X-CAD: Optimizing CAD Models with Extended Finite Elements

CHRISTIAN HAFNER, IST Austria
CHRISTIAN SCHUMACHER, Disney Research
ESPEN KNOOP, Disney Research
THOMAS AUZINGER, IST Austria
BERND BICKEL, IST Austria
MORITZ BÄCHER, Disney Research

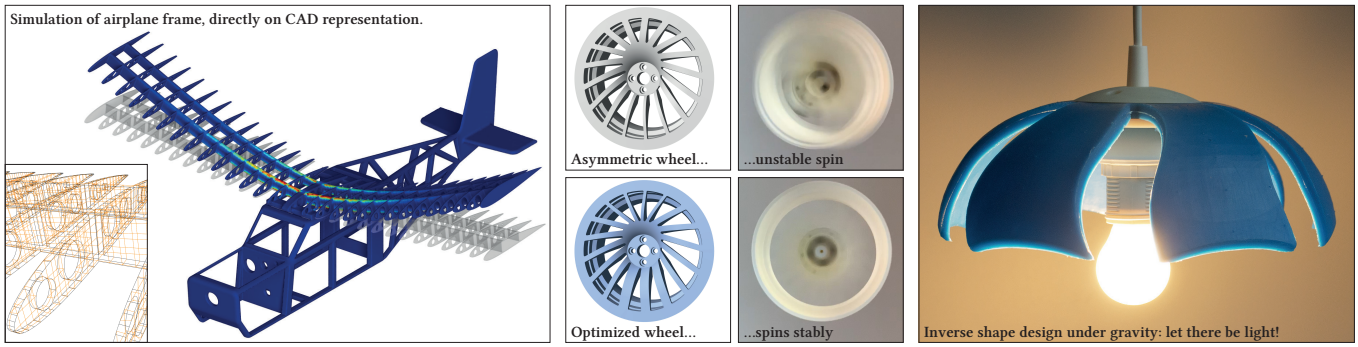


Fig. 1. We present a differentiable deformable solid simulation (left) that enables shape optimization on CAD representations (middle, right) while preserving a model's manufacturability, function, and appearance. We demonstrate our optimization on a range of objectives including co-optimization of strength-to-weight ratio and mass distribution (middle), and rest shape optimization (right).

We propose a novel generic shape optimization method for CAD models based on the eXtended Finite Element Method (XFEM). Our method works directly on the intersection between the model and a regular simulation grid, without the need to mesh or remesh, thus removing a bottleneck of classical shape optimization strategies. This is made possible by a novel hierarchical integration scheme that accurately integrates finite element quantities with sub-element precision. For optimization, we efficiently compute analytical shape derivatives of the entire framework, from model intersection to integration rule generation and XFEM simulation. Moreover, we describe a differentiable projection of shape parameters onto a constraint manifold spanned by user-specified shape preservation, consistency, and manufacturability constraints. We demonstrate the utility of our approach by optimizing mass distribution, strength-to-weight ratio, and inverse elastic shape design objectives directly on parameterized 3D CAD models.

CCS Concepts: • **Theory of computation → Continuous optimization**; • **Computing methodologies → Modeling and simulation**; **Shape modeling**; **Parametric curve and surface models**.

Additional Key Words and Phrases: Shape Optimization, CAD Processing, XFEM, Simulation

ACM Reference Format:
Christian Hafner, Christian Schumacher, Espen Knoop, Thomas Auzinger, Bernd Bickel, and Moritz Bächer. 2019. X-CAD: Optimizing CAD Models with Extended Finite Elements. *ACM Trans. Graph.* 38, 6, Article 157 (November 2019), 15 pages. https://doi.org/10.1145/3355089.3356576

Authors' addresses: Christian Hafner, IST Austria; Christian Schumacher, Disney Research; Espen Knoop, Disney Research; Thomas Auzinger, IST Austria; Bernd Bickel, IST Austria; Moritz Bächer, Disney Research.

1 INTRODUCTION

Ever since Ivan Sutherland laid the foundation of modern Computer-Aided Design (CAD) with his revolutionary computer program Sketchpad [Sutherland 1963], CAD systems have become a core pillar of innovation. In combination with simulation, they have empowered us to design architectural masterpieces like the Sidney Opera House, or fuel-efficient airplanes like the Airbus A350 XWB or the Boeing 787-9 Dreamliner. Yet, it remains onerous to treat CAD model parameters as design variables in optimizations.

In modern CAD systems, a *boundary* representation (B-rep), predominantly composed of Non-Uniform Rational Basis Spline (NURBS) patches, is used to describe solid models. The success of B-rep is attributed to the many desirable properties of NURBS, enabling the precise representation of analytical and free-form shapes, and modeling operations such as extrusion, chamfering, or blending. While advantageous for manual design, strength-to-weight or rest shape optimization require the solution of a Partial Differential Equation (PDE) on the enclosed *volume*.

Although progress has been made in isogeometric analysis, where PDEs are solved on volumetric NURBS representations, the generation of volumetric NURBS for general B-rep input is highly challenging [Cottrell et al. 2009]. Hence, it is still the de facto standard to solve PDEs on a volumetric *mesh* representation. However, because shape optimization requires a *differentiable* simulator, and even moderate changes to design variables demand repeated conversion

and remeshing, the use of CAD in combination with optimization is limited.

In this paper, we propose a novel differentiable deformable solid simulation that enables generic shape optimization directly on CAD models. To dodge remeshing discontinuities and avoid a dependence of shape derivatives on the simulation mesh, we intersect the CAD model with a regular hexahedral grid that we keep *constant* throughout optimizations. While the resulting simulation mesh is not conformal, we enrich elements that are cut by the B-rep, representing the solid-void boundary *explicitly*. To accurately integrate over the subvolumes of the resulting extended finite elements, we adopt and extend a recent quadrature scheme [Müller et al. 2013].

CAD models are often tailored for fabrication using a particular manufacturing technology. For example, if we target casting or injection molding, a model has to be undercut-free and observe a minimal draft angle constraint. To preserve a model's manufacturability, function, and appearance during optimization, we therefore analyze properties of neighboring NURBS patches, and provide the user with the option of defining constraints on the mapping of high-level shape to low-level patch parameters.

We demonstrate the efficacy, generality, and utility of our technique by minimizing common strength-to-weight, rest shape, and mass distribution objectives on a set of complex CAD models with a plethora of thin and sharp features. With a set of validation experiments, we further show that our XFEM simulation results are in excellent agreement with simulations performed on a conformal mesh with standard FEM, for both linear elasticity and hyperelasticity problems.

Succinctly, we propose and contribute

- a *differentiable* simulator that enables generic shape optimization on CAD models.
- an extension of a hierarchical quadrature scheme [Müller et al. 2013] to accurately and reliably integrate subelement detail of varying shape and size.
- a change of basis for enriched elements, making it straightforward to turn a standard FEM into an efficient XFEM implementation.
- a differentiable projection of shape parameters onto a set of shape, function, and manufacturability constraints, and efficient shape derivatives of our hierarchical quadrature.

2 RELATED WORK

Fabrication-Aware Shape Optimization. Recent years have witnessed an increasing interest in fabrication-aware shape optimization that targets the design of shapes with desired properties. Examples include mass distribution optimization to make a model stably stand [Prévost et al. 2013], spin [Bächer et al. 2014], or float [Wang and Whiting 2016]; sound spectra optimization with applications in metallophone design [Bharaj et al. 2015]; optimizing the strength-to-weight ratio of printed parts [Stava et al. 2012; Zhou et al. 2016]; or rest shape optimization for nonlinear elastic objects [Chen et al. 2014; Skouras et al. 2012]. Although we share the goal of devising a generic shape optimization with Musialski et al. [2016], our focus is on the optimization of CAD models instead of meshed representations.

Analyzing CAD Models. The predominant approach for analyzing CAD models is to first volumetrically mesh the enclosed domain, then discretize PDEs over finite elements [Hu et al. 2018; Schneider et al. 2018]. However, whenever a CAD model parameter is adjusted, the conforming representation has to be refined and remeshed. In addition to computational overhead, changes in the discretization can also lead to popping artifacts and non-smooth objective functions during shape optimization. In the last decade, the field of *IsoGeometric Analysis* (IGA) has arisen. IGA attempts to unify design and analysis by augmenting the 2D surface NURBS with 3D solid NURBS elements. This tight integration between CAD and analysis offers tremendous advantages in shape optimization, but it comes with the challenge of identifying a parameterization of the volumetric domain from its boundary, currently an active area of research [Daxini and Prajapati 2017; Hsu et al. 2015]. Mesh-free simulation methods [Martin et al. 2010; Müller et al. 2004; Nayroles et al. 1992] generate approximations of the deformation field from only a set of points, but these methods face the challenge that accurately representing complex features requires an adequate sampling of a model's boundary. Similar to traditional mesh-based approaches, this raises sampling and segmentation issues [Faure et al. 2011]. Trying to maintain the advantages of mesh-free methods while addressing these shortcomings, the enrichment concept of the XFEM enables the introduction of additional degrees of freedom (DOFs) for elements cut by the boundary. This enables reduction of the complexity of the geometric meshing domain [Fries and Belytschko 2010]. The method was initially developed by Belytschko and Black [1999] for modeling crack growth. In graphics, the method has received attention for simulating the cutting of shells and solids [Jeřábková and Kuhlen 2009; Kaufmann et al. 2009; Koschier et al. 2017].

In this work, our strategy is to build on the XFEM method in order to avoid complex meshing and remeshing operations. In doing so, we face the technical challenge of accurately integrating on domains delimited by NURBS surfaces. Safardi et al. [2015; 2016] suggested to utilize NURBS to augment the finite element approximation space and minimize geometric errors associated with the discretization of a complex domain in combination with the generalized finite element method. Haasemann et al. [2011] developed a quadratic finite element formulation based on the XFEM and NURBS, which was later extended to higher order approximations [Legrain 2013]. In these works, it is required that the boundary of a model cuts an element edge of the nonconforming mesh at most once, and elements are cut into at most two parts. To fulfill these requirements, elements are recursively subdivided, resulting in discontinuous changes of element domains during shape optimization. A conceptually similar strategy, which also faces similar challenges, is to simplify the integration domain by subdividing elements that are cut by boundaries into curved quadrilaterals and triangles [Kudela et al. 2015]. However, so far this method has been demonstrated for 2D domains only, and a robust extension to 3D is challenging because of non-trivial configurations emerging from boundaries with complex subelement detail. Alternatively, Müller et al. [2017; 2013] showed how to integrate over volumes and surfaces defined by implicitly-given level sets, but their integration strategy may fail when surface features smaller than a simulation element are present. In our work, we propose quadrature rules that solve this problem.

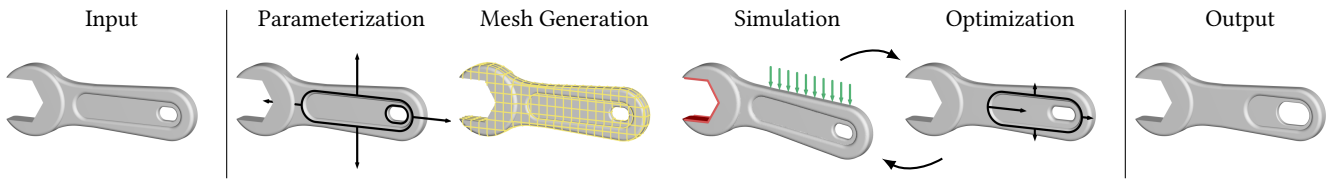


Fig. 2. **Shape Optimization on CAD** Given a CAD model (Input), a user first selects shape parameters for optimization (Parameterization). Intersecting the model with a regular hexahedral mesh (Mesh Generation), we combine a novel integration scheme with XFEM to integrate finite element quantities with subelement precision (Simulation). Analytical shape derivatives of model-grid intersections, our integration scheme, and XFEM simulations (Optimization), enable function-, manufacturability-, and appearance-preserving shape optimization of CAD models (Output).

Optimizing CAD Models. As topology optimization methods do not impose restrictions on attainable shapes [Bendsøe and Sigmund 1999; Liu et al. 2018], the resulting shapes often cannot be directly fabricated and need to be reinterpreted as CAD models. In contrast, shape optimization introduces a limited set of design variables, and the design problem is formulated directly on the parameterized CAD representation. In practice, shape optimization of CAD models is known to be fragile and delicate to use because of different representations in design and analysis. An extensive review of parametric shape optimization techniques can be found in [Daxini and Prajapati 2017].

The potential of using XFEM for shape optimization has already been highlighted by Duysinx et al. [2006], combining simple parametric features such as circles, ellipses, and squares with a level set topology optimization using XFEM. Zehnder et al. [2017] use a similar approach to optimize a set of spherical inclusions. However, these techniques rely on the assumption that boundaries can be well-approximated by functions which are linear on elements. Therefore, they cannot capture features that are common in CAD models.

Recently, Najafi et al. [2017] introduced an optimization scheme built on NURBS-based interface-enriched generalized FEM (IGFEM), illustrating the application for materials with inclusions. While conceptually we share a similar vision, our method differs in several ways: Firstly, the NURBS-enhanced IGFEM handles problems with weak discontinuities, such as different material interfaces; our method can handle strong discontinuities, such as cuts. Secondly, their method is described and was implemented and validated only for 2D shape optimization problems. Finally, an extension of their integration strategy to 3D would require a tessellation of the volume with 3D NURBS elements, resulting in similar challenges as faced by other IGA methods. We demonstrate the applicability of our method to complex 3D CAD models.

Notably, taking a data-driven approach, Schulz et al. [2017] proposed an interactive design exploration for CAD models in which an analysis is precomputed for parameter samples on an adaptive grid and then interpolated during run-time. Although this approach is promising for shapes with just a few design parameters, the combinatorial complexity becomes prohibitive for higher-dimensional problems.

3 OVERVIEW

Before we delve into our technical contributions, we provide a high-level overview of how we simulate the elastic response of a CAD model, and optimize shape parameters with respect to objectives that depend on this response.

3.1 CAD Model Representation

The most general form of a CAD model that we consider is a closed NURBS mesh, i.e., a set of NURBS patches that form a C^0 surface. We assume that the modeler applied appropriate engineering judgment during initial design, so that the input fulfills *geometric* requirements for manufacturing (see Fig. 2, Input).

We rely on *projective* coordinates to represent NURBS patches, where points $[x, y, z]^T$ in Euclidean coordinates are represented with points $[wx, wy, wz, w]^T$ in projective space $\mathbb{P}^3$. We therefore assume a NURBS patch with control points $\mathbf{q}_{i,j} \in \mathbb{P}^3$ and polynomial basis functions $B_{i,j} : \mathbb{R}^2 \to \mathbb{R}$ to be a parametric mapping

$$\boldsymbol{\sigma} : \mathbb{R}^2 \to \mathbb{P}^3 \qquad \mathbf{u} \mapsto \sum_{i,j} B_{i,j}(\mathbf{u})\, \mathbf{q}_{i,j} \tag{1}$$

from uv-coordinates $\mathbf{u} = [u, v]^T$ to a point $\boldsymbol{\sigma}(\mathbf{u})$ in projective coordinates. In contrast to the rational form $\hat{\boldsymbol{\sigma}} : \mathbb{R}^2 \to \mathbb{R}^3$ in Euclidean space, this form is more convenient for simulation and optimization because $\boldsymbol{\sigma}$ is *polynomial*. We use this definition everywhere, and recover Euclidean coordinates by perspective division where necessary.

During optimizations, we seek to ensure that a model remains manufacturable, and that changes to shape parameters do not negatively impact its function or characteristic appearance. For instance, in CAD, it is commonplace to round off sharp edges and corners of models by introducing fillets. If we moved control points of patches in an uncontrolled manner, we could easily reintroduce sharp features between neighboring patches.

To prevent undesirable changes to the model, we put users in control, letting them define an implicit mapping from high-level shape parameters $\mathbf{p}$ to the set of m control points $\mathbf{q} \in \mathbb{R}^{4m}$ of the NURBS mesh

$$\mathbf{c}_{\text{para}}(\mathbf{p}, \mathbf{q}(\mathbf{p})) = 0. \tag{2}$$

During optimizations, we then enforce these constraints $\mathbf{c}_{\text{para}}$, keeping the number and topology of patches fixed. We defer a detailed discussion of our *parameterization* until Sec. 6.

A key benefit of our technique is that structural or related objectives are directly minimized on a CAD representation, and the optimized output (Fig. 2, Output) can be loaded into a modeling tool for further refinement, or to design the mold for manufacturing by casting or modeling.

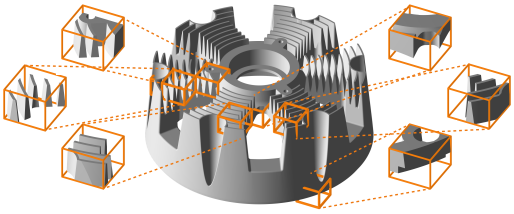


Fig. 3. **Hexahedral Meshing** If we embed a CAD model in a regular hexahedral simulation mesh, the NURBS representation is cut into arbitrarily complex subvolumes near the boundary.

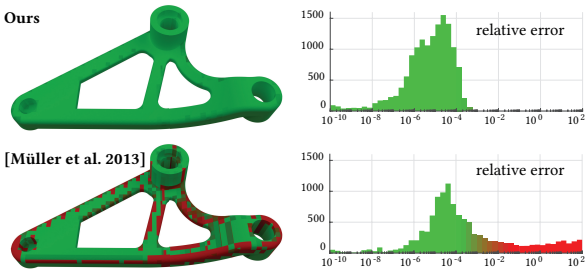


Fig. 4. **Quadrature** For detailed subvolumes resulting from intersections of CAD models with regular simulation meshes, Müller et al.'s quadrature [2017; 2013] (bottom) introduces large numerical errors while ours (top) is accurate. To generate the histograms on the right, we integrated the polynomial basis that is used for rule construction. To compute the relative error, we divided the absolute value of the difference of the exact and numerical integrals over the subvolume, by the absolute value of the exact integral over the element volume.

3.2 Deformable Solid Simulation on CAD

We target shape optimization of CAD representations where objectives depend on the elastic response of the material delimited by the boundary representation. To achieve this goal, our simulation has to be sufficiently *smooth* and *differentiable*. A standard conformal Finite Element (FE) discretization is ill-suited here because, if the shape of a model undergoes significant changes, remeshing is unavoidable. These uncontrolled topological changes lead to discontinuities, and therefore to a non-differentiable simulation.

To mitigate this problem, we propose to *embed* the CAD model in a regular hexahedral simulation mesh (Fig. 2, Mesh Generation). This mesh remains *constant* during optimization. To compute the elastic response $\mathbf{x} \in \mathbb{R}^{3n}$ of a CAD model (Fig. 2, Simulation), we seek to minimize the standard potential energy

$$E(\mathbf{x}) = E_{\text{int}}(\mathbf{x}) - E_{\text{ext}}(\mathbf{x}) \quad \text{with} \quad E_{\text{int}}(\mathbf{x}) = \int_V \Psi(\mathbf{x}, \mathbf{X})\, \mathrm{d}V, \qquad (3)$$

where we integrate the material-dependent strain energy density Ψ [Sifakis and Barbič 2012] over points $\mathbf{X} \in \mathbb{R}^3$ in the undeformed volume V. The result of this minimization is a static equilibrium $E_{\mathbf{x}} = 0$ ($E_{\mathbf{x}}$ abbreviates the partial derivative $\frac{\partial E}{\partial \mathbf{x}}$) where the internal or elastic forces $E_{\text{int},\mathbf{x}}$ are in balance with external forces $E_{\text{ext},\mathbf{x}}$. However, in contrast to standard FEM, the volume V enclosed in the CAD model is the intersection of the B-rep with a regular hexahedral mesh, and elements on the boundary are cut into arbitrarily complex subvolumes as we illustrate in Fig. 3.

To represent the solid-void boundary in cut elements, *implicit* descriptions where signed distances to the boundary are discretized at mesh nodes, are common (see, e.g., [Schumacher et al. 2018]). However, they fail to resolve subelement detail.

To loosen the coupling between mesh resolution and simulation precision, we therefore represent cuts in elements *explicitly* with an enrichment, and devise a quadrature scheme that integrates quantities such as the elastic energy E_{int} over complex subvolumes reliably and accurately. Although Koschier et al. [2017] recently addressed a related problem for simulations of detailed cuts by building on the quadrature scheme by Müller et al. [2013], a direct extension of their technique to our setting is not possible because integration over detailed curved domains with features smaller than a simulation element lead to large numerical errors as we epitomize in Fig. 4 bottom.

Based on this observation, we propose

(1) a modified set of quadrature rules that accurately handle integration over curved domains of varying shape and size, delimited by NURBS and planar patches (see Fig. 4 top).
(2) a refinement of rule construction to significantly reduce the cost of evaluations of shape derivatives and updates to rules when shape parameters change.
(3) a change of basis for enriched shape functions that makes it straightforward to turn standard FEM into efficient XFEM implementations.

We discuss our integration scheme in Sec. 4, and our XFEM formulation in Sec. 5.

3.3 Optimizing CAD Models

A first generic type of objective we seek to optimize integrates a function g that depends on the elastic response of the model over the volume enclosed by the B-rep

$$f(\mathbf{q}(\mathbf{p}), \mathbf{x}(\mathbf{p})) = \int_{V(\mathbf{q})} g(\mathbf{x}, \mathbf{X})\, \mathrm{d}V. \qquad (4)$$

Because the control points of the B-rep define the volume V, and changes to shape parameters translate to changes in control points, the rest shape of the model, hence also its elastic response, *implicitly* depend on the shape parameters $\mathbf{p}$.

Recent examples where this type of objective is used are traditional compliance optimization (see, e.g., [Liu et al. 2018]) or the minimization of the potential of failure of structures [Schumacher et al. 2018]. For compliance optimization, the strain energy density Ψ for linear elasticity is integrated over the rest volume, and for the minimization of the potential failure, a metric that measures the exponentiated distance of the Cauchy stress to the failure surface of a generic failure criterion, is integrated.

It is often desirable to minimize objectives that depend on the elastic response together with mass distribution objectives. For example, if we seek to optimize the strength-to-weight ratio of an asymmetric wheel design (compare with Figs. 1 and 15), the center of mass has to lie on the wheel's axis, and the major axis of the

moment of inertia has to align with this axis [Bächer et al. 2014]. Otherwise, the model cannot fulfill its function.

To support the co-optimization of such combinations of objectives, we introduce a second type of objective that integrates standard functions over the volume delimited by the B-rep

$$f(\mathbf{q}(\mathbf{p})) = \int_{V(\mathbf{q})} g(\mathbf{X})\,\mathrm{d}V. \tag{5}$$

Substituting an either constant or spatially-varying density $\rho(\mathbf{X})$ times a monomial $t \in \{1, X, Y, Z, XY, XZ, YZ, X^2, Y^2, Z^2\}$ for the integrand, we can compute a model's mass, its center of mass, and moment of inertia [Bächer et al. 2014]. For example, if we integrate the density (times the constant 1), we get the mass of a CAD model, and combined with our first type, we can formulate common strength-to-weight ratio optimizations.

A third type of objective we seek to support only depends on the elastic response of a model

$$f(\mathbf{x}(\mathbf{p})) = g(\mathbf{x}). \tag{6}$$

This type of objective enables, for example, inverse shape design [Chen et al. 2014], where the rest shape of the model is optimized such that the deformed model matches a target shape under a predefined load as closely as possible.

Shape Optimization. In our shape optimizations (Fig. 2, Optimization; Sec. 6), we then seek to minimize a single or a weighted combination of these objectives over the parameterized volume that a CAD model encloses

$$\min_{\mathbf{p}} f(\mathbf{p}, \mathbf{q}(\mathbf{p}), \mathbf{x}(\mathbf{p})) \quad \text{s.t.} \quad \begin{matrix} \mathbf{c}_{\text{para}}(\mathbf{p}, \mathbf{q}(\mathbf{p})) = 0 \\ E_{\mathbf{x}}(\mathbf{q}(\mathbf{p}), \mathbf{x}(\mathbf{p})) = 0 \end{matrix}, \tag{7}$$

enforcing first-optimality constraints on our parameterization and the elastic response. To prevent shape parameters from taking on values that would lead to non-manufacturable designs, we add an additional term to f that directly depends on $\mathbf{p}$ (e.g., penalizing the radii of two cylinders to prevent them from overlapping), hence the direct dependence of f on $\mathbf{p}$.

To enable shape optimization on CAD, we contribute

(1) a *continuous* projection of shape parameters onto the constraint manifold spanned by the user-specified parameterization, guaranteeing well-posedness of the problem, and
(2) a technique to *efficiently* compute derivatives of our hierarchical quadrature rules.

In Sec. 7, we demonstrate our technique on a wide range of examples including the compliance minimization of a motor housing, the inverse shape design of a lampshade, and the co-optimization of the strength-to-weight ratio and balance of an asymmetric wheel.

4 INTEGRATING OVER SUBVOLUMES

To simulate a complex CAD model on a simulation mesh that is *not* conformal, we require quadrature rules for integration of functions over (1) subvolumes, which are part of the model interior (e.g., to accumulate elastic force density), and over (2) regions on the model surface (e.g., to aggregate surface traction).

In our setting, quadrature rules are not readily available because the integration domains are generated at runtime as intersections between arbitrary models and planes (see Fig. 3). Müller et al. [2013] proposed a moment-fitting technique to compute on-the-fly quadrature rules for domains with curved boundaries. Originally developed for integration of implicit representations and fluid simulation, we propose a variant that avoids failure cases when integrating over *explicitly* defined, detailed subvolumes (compare with Fig. 4), and—at the same time—increases computational efficiency of the technique.

In the following, we assume that the NURBS patches have been cut along edges and faces of the hexahedral mesh. To do so, we rely on *robust* algebraic curve tracing [Bajaj et al. 1988], and perturb grid planes to avoid corner cases. Refer to Fig. 5 for an example case of a patch-element intersection (left): Analogously to Müller et al. [2013], we build integration schemes in a hierarchical manner. We use edge and curve rules (right) to integrate over areas and surfaces, and area and surface rules (middle) to integrate over volumes (left).

Below, we use $g : \mathbb{R}^3 \to \mathbb{R}$ to denote a general function defined on the volumetric domain enclosed by the CAD model. Our goal is to construct quadrature rules that *exactly* integrate g, drawn from a function space spanned by a set of basis functions, over a domain D

$$\int_D g(\mathbf{X})\,\mathrm{d}D = \sum_j w_j g(\mathbf{X}_j). \tag{8}$$

The domain D is either one-dimensional, for integration along axis-aligned edges E (in blue in Fig. 5) or curves C (in yellow), two-dimensional, for integration over planar patches A (in light gray) or curved surfaces S (in dark gray), or three-dimensional, for integrals over volumetric domains V. Because we integrate over undeformed domains, we use capital $\mathbf{X}_j \in \mathbb{R}^3$ to refer to quadrature points corresponding to weights w_j.

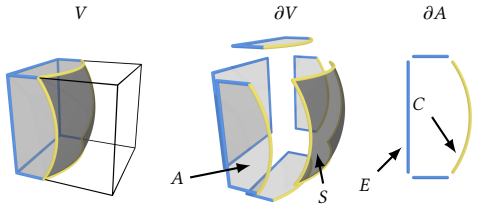


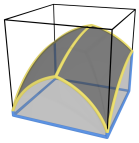
Fig. 5. **Nesting of Integration Rules** To construct rules for integration over volumes V (left), we rely on rules for integration over the volume's boundary ∂V, decomposed into planar areas A (in light gray) and curved surfaces S (in dark gray). To generate rules for area and surface integrals (middle), we express integrals along the boundary ∂A with integrals along edges E and curve segments C (right).

4.1 Integrating along Edges and Curves

To integrate along axis-aligned edge segments E, we form one-dimensional integrals, for example $\int_{[a,b]} g(X, Y, Z)\,\mathrm{d}X$ for an integral along the X-axis. Like Müller et al. [2013], we perform a change of variables to map the interval $[a, b]$, delimited by the two segment endpoints, to the interval $[0, 1]$, then apply a standard Gauss-Legendre rule.

Intersections of NURBS patches with hexahedral elements form *planar* curves that are embedded in planes parallel to one of the coordinate planes. To integrate along these curves, Müller et al. [2013] construct a divergence-free basis, and express curve integrals with

sums of integrals along straight edge segments (see supplemental material for detail). However, as we illustrate in Fig. 6 with a circle example, their method loses accuracy as the chord length (in yellow) of the cut circle decreases. Another case that arises in our application domain is that patches intersect within elements, and form, in general, *spatial* curves as shown in the inset.



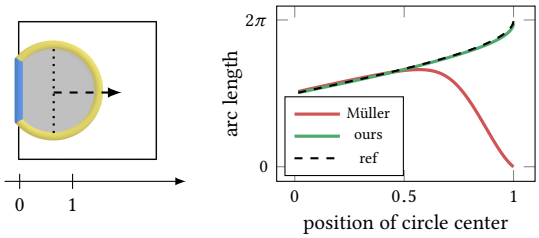


Fig. 6. **Integrating Features** Designed for integrating over implicitly defined domains, Müller et al.'s method [2013] fails to integrate subelement detail as we demonstrate with a curve integral to compute the arc length of a cut circle (in yellow). While our integration accurately predicts the analytical arc length (green vs. dashed line), Müller et al.'s method is less accurate (see red line) the shorter the edge E (in blue) becomes.

For accurate integration along curves C, we parameterize them with a mapping from $t \in [a, b]$ to spatial curve points $\mathbf{r}(t)$, then use a Gauss-Legendre rule for numerical integration of the transformed integrals $\int_{[a,b]} g(\mathbf{r}(t)) \|\mathbf{r}'(t)\| \, \mathrm{d}t$ where $\mathbf{r}'$ denotes the derivative of the mapping with respect to parameter t.

While we can expect intersection curves to be sufficiently smooth, we cannot, in general, extract analytical parameterizations from intersections of the B-rep with the hexahedral mesh. Hence, we represent them with sample points, and approximate its parametric form with a Lagrange interpolating polynomial as we describe in more detail in our supplemental material. Note that the accuracy of the Gauss-Legendre integration is preserved for a polynomial interpolation of sufficiently high degree [Atkinson and Venturino 1993].

4.2 Integrating over Areas and Surfaces

To integrate over planar areas and surfaces, we make use of the first nesting of our hierarchical integration scheme. There are two cases to consider: (1) integrals over *planar areas* that lie in grid planes of the simulation mesh, and (2) integrals over *curved surfaces*, represented by NURBS patches.

Area Integrals. For integration over planar domains A that are parallel to one of the coordinate planes, we use moment fitting analogously to Müller et al. [2013]. Moment fitting is similar to the construction of Newton-Cotes rules: given a set of predefined quadrature points, a system of equations is solved to compute corresponding quadrature weights such that a polynomial basis $\{p_1, \ldots, p_m\}$, spanned by a set of m functions, is integrated *exactly*. However, a crucial difference is that, due to the non-standard domain, basis functions cannot be integrated *analytically*, and this is the reason why a nesting is necessary.

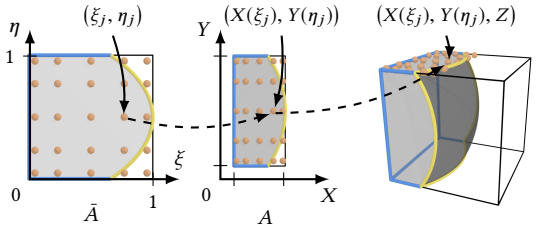


Fig. 7. **Integrating over Areas** To integrate over planar domains A (right), we first compute an axis-aligned bounding box (middle), then transform the integral to the isoparametric domain $\bar{A}$ (left). In the domain $\bar{A}$, we use standard Gauss quadrature points (ξ_j, η_j) (in beige), and transform them back to spatial $(X(\xi_j), Y(\eta_j), Z)$ after rule construction. For surface integrals, we perform this transformation in parameter domain.

There is a second important difference between Newton-Cotes and Müller et al.'s construction of rules: To avoid having to rederive rules whenever the domain $[a, b]$ changes, Newton-Cotes rules are constructed and tabulated for standard ranges (e.g., the range $[0, 1]$). For moment fitting, tabulation of rules is not possible due to the non-standard domains. However, transforming the area integrals such that the bounding boxes of the transformed domains coincide with the unit square $[0, 1]^2$ enables the use of a *single* system matrix for the construction of *all* our area and surface integrals. This bears far-reaching advantages: Whenever we make changes to shape parameters, rules have to be updated. If the system matrix does *not* depend on the parameters, the system can be prefactorized and rules updated more efficiently. In addition, we can avoid having to take derivatives of the inverse of a matrix, leading to a significant performance increase for evaluations of shape derivatives.

More formally, to integrate over an area A that lies in the XY-plane, we first compute an axis-aligned bounding box $[a, b] \times [c, d]$ (compare with Fig. 7 middle), then define a mapping between isoparametric variables ξ and η and the two spatial coordinates

$$\begin{pmatrix} X(\xi) \\ Y(\eta) \end{pmatrix} = \begin{pmatrix} b-a & \\ & d-c \end{pmatrix} \begin{pmatrix} \xi \\ \eta \end{pmatrix} + \begin{pmatrix} a \\ c \end{pmatrix}. \tag{9}$$

Because of the linearity of the mapping, its Jacobian is constant and the transformed integral reads

$$\int_{\bar{A}} g(X(\xi), Y(\eta), Z) \det\left(\frac{\partial(X, Y)}{\partial(\xi, \eta)}\right) \mathrm{d}\bar{A} \tag{10}$$

where $\bar{A}$ is the non-uniformly scaled domain, and the determinant of the Jacobian is set to the constant $(b-a)(d-c)$.

Moment Fitting. For readers unfamiliar with moment fitting, we provide a brief technical overview here, pointing the interested reader to the original work [Müller et al. 2013] or our supplemental material for detail.

To compute the quadrature weights, we form the system

$$\underbrace{\begin{pmatrix} p_1(\xi_1, \eta_1) & \ldots & p_1(\xi_n, \eta_n) \\ \vdots & \ddots & \vdots \\ p_m(\xi_1, \eta_1) & \ldots & p_m(\xi_n, \eta_n) \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} \bar{w}_1 \\ \vdots \\ \bar{w}_n \end{pmatrix}}_{\mathbf{w}} = \underbrace{\begin{pmatrix} \int_{\bar{A}} p_1(\xi, \eta) \, \mathrm{d}\bar{A} \\ \vdots \\ \int_{\bar{A}} p_m(\xi, \eta) \, \mathrm{d}\bar{A} \end{pmatrix}}_{\mathbf{b}}$$

with constant matrix $\mathbf{A}$, evaluating the basis functions at Gauss quadrature points [Müller et al. 2013]. For moment fitting to work, there need to be at least $n \geq m$ quadrature points, (ξ_j, η_j), forming an underdetermined system. To solve the system, we factorize the pseudo-inverse in the minimal-norm solution

$$\mathbf{w} = \mathbf{A}^T \left(\mathbf{A}\mathbf{A}^T\right)^{-1} \mathbf{b}. \tag{11}$$

While matrix $\mathbf{A}$ is independent of the integration domain, the right-hand side is not. To evaluate $\mathbf{b}$, we make use of the divergence theorem

$$\int_{\bar{A}} p_i(\xi,\eta)\,\mathrm{d}\bar{A} = \int_{\partial\bar{A}} \mathbf{n}(\xi,\eta)\cdot\mathbf{P}_i(\xi,\eta)\,\mathrm{d}\bar{s} \tag{12}$$

where $\mathbf{n}$ is the outward-facing normal at (ξ,η) and

$$\mathbf{P}_i(\xi,\eta) = \frac{1}{2}\begin{pmatrix} \int p_i(\xi,\eta)\,\mathrm{d}\xi \\ \int p_i(\xi,\eta)\,\mathrm{d}\eta \end{pmatrix} \tag{13}$$

the antiderivative, chosen such that $\nabla\cdot\mathbf{P}_i = p_i$. Note that the boundary of the domain $\partial\bar{A}$ consists of straight edge segments and planar curves (see Fig. 5 right), and we use rules developed in Sec. 4.1 to numerically integrate along them.

After construction, we transform the weights and quadrature points back to the original domain

$$w_j = (b-a)(d-c)\bar{w}_j \quad \text{and} \quad \mathbf{X}_j = \left[X(\xi_j), Y(\eta_j), Z\right]^T. \tag{14}$$

Surface Integrals. For surface integrals, Müller et al. [2013] proceed analogously to the edge-curve case, and express surface integrals with a sum of integrals over planar areas. The resulting rules suffer from similar issues as the one-dimensional rules: if the planar areas become too small, the integration error increases uncontrollably.

For accurate integration over surfaces, we instead make use of the parametric form of NURBS patches, expressing them as *area* integrals in parameter space

$$\int_S g(\mathbf{X})\,\mathrm{d}S = \int_A g(\hat{\boldsymbol{\sigma}}(u,v))\,\|\hat{\boldsymbol{\sigma}}_u(u,v)\times\hat{\boldsymbol{\sigma}}_v(u,v)\|\,\mathrm{d}A. \tag{15}$$

Thus, integration weights are computed in uv-space, then transformed to physical coordiantes by multiplication with the area factor $\|\hat{\boldsymbol{\sigma}}_u\times\hat{\boldsymbol{\sigma}}_v\|$. Due to the non-linearity of this transformation, the resulting rule may not exactly integrate polynomials in physical coordinates. However, this error is mitigated by the smoothness of the area factor as we empirically show in Fig. 4.

4.3 Integrating over Volumes

To integrate over volumes, we proceed analogously to area integration (compare with Fig. 8) as we formally describe in our supplemental material: We first compute a bounding box $[a,b]\times[c,d]\times[e,f]$, and define a mapping to the unit cube. To evaluate the integrals of basis functions over the transformed domains, we use area and surface rules developed in the previous section, establishing a second and final layer of nesting.

An important detail is that the non-uniform scaling $\mathbf{S} = \mathrm{diag}(b-a, d-c, f-e)$ has to be taken into account when we transform integrals over curved domains as we illustrate in Fig. 8. We use the rule for linear transformations of cross products to account for this scaling in our surface integrals

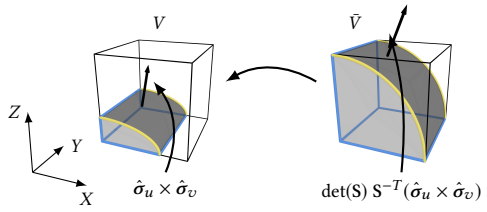


Fig. 8. **Integrating over Volumes** To integrate over a volumetric domain V, we transform an axis-aligned bounding box of the volume to the unit cube $\bar{V}$. For consistency, normals $\hat{\boldsymbol{\sigma}}_u \times \hat{\boldsymbol{\sigma}}_v$ on curved surfaces need to be transformed before we can apply surface area rules. We use the linear transformation rule for cross products to do so.

$$\int_A g(\mathbf{X})\,\|\mathbf{S}\hat{\boldsymbol{\sigma}}_u\times\mathbf{S}\hat{\boldsymbol{\sigma}}_v\|\,\mathrm{d}A = \int_A g(\mathbf{X})\det(\mathbf{S})\,\|\mathbf{S}^{-T}\hat{\boldsymbol{\sigma}}_u\times\hat{\boldsymbol{\sigma}}_v\|\,\mathrm{d}A. \tag{16}$$

4.4 Polynomial Bases and their Degree

The choices of bases for the construction of curve, area, and volume rules are not independent due to two reasons: firstly, the curve or area rules are used to evaluate the flux of polynomials $\mathbf{P}_i$ in the construction of area and volume rules, respectively. Secondly, we use surface rules to evaluate polynomials in spatial coordinates arising from the finite element method, and therefore integrals over $g\circ\hat{\boldsymbol{\sigma}}$ with polynomial g need to be approximated sufficiently well. These observations inform our choice of basis $\{\xi^i\eta^j\zeta^k : i+j+k \leq 4\}$ for our volume rules, and the compatible basis $\{\xi^i\eta^j : 0 \leq i,j \leq 4\}$ for our area rules. For surface rules, we use maximal degree of 5 instead of 4 to account for the additional nonlinearity in the area factor.

5 SIMULATING CUT ELEMENTS

To illustrate the use of our rules in solving static elasticity problems on domains enclosed by a B-rep, it remains to discuss how we can accurately represent *interpolated* quantities on an *unfitted* mesh. To this end, we enrich elements which are cut by the boundary.

After discussing standard elements, we highlight how a change of basis for enriched elements (1) enables the use of a *standard* FEM implementation for element energy, force, and tangent stiffness evaluations for cut elements, (2) increases the efficiency of these evaluations, and (3) preserves desirable properties of Lagrange shape functions.

Although we apply our technique to elasticity problems within the scope of this paper, our technical contributions are not limited to a particular PDE.

5.1 Standard Elements and Elasticity

Because our simulation mesh consists of regular cuboids, hexahedral elements lend themselves. We use standard Lagrange shape functions $N_i : \mathbb{R}^3 \to \mathbb{R}$ to interpolate an element's undeformed nodes $\mathbf{X}_i$, defining a mapping $\mathbf{X}(\boldsymbol{\xi}) = \sum_i N_i(\boldsymbol{\xi})\mathbf{X}_i$ from natural to physical coordinates. Relying on the same interpolation for the

deformed configuration, we define the deformation gradient

$$\mathbf{F}(\boldsymbol{\xi}) = \frac{\partial \mathbf{x}(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} \left(\frac{\partial \mathbf{X}(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} \right)^{-1} \quad \text{with} \quad \mathbf{x}(\boldsymbol{\xi}) = \sum_i N_i(\boldsymbol{\xi}) \mathbf{x}_i. \tag{17}$$

The choice of bases for rule construction depends on the order of our shape functions. It is worth pointing out that the mapping from natural to physical coordinates is *linear* for hexahedral elements, meaning that only the coefficients of the polynomial shape functions change. However, note that, unlike for tetrahedral elements, the deformation gradient is *not* constant for linear N_i.

To compute a static equilibrium, the strain energy density $\Psi(\mathbf{F})$ of a linear or hyperelastic material [Sifakis and Barbič 2012; Sin et al. 2013] is integrated over all hexahedral elements of volume V_e

$$E_{\text{int}}(\mathbf{x}) = \sum_e \int_{V_e} \Psi(\mathbf{F}(\boldsymbol{\xi}))\, \mathrm{d}V \tag{18}$$

where the integral over element e only depends on the incident nodal degrees of freedom.

Note that we perform integration in physical and *not* in natural coordinates, hence we transform quadrature points $\mathbf{X}_j$ in the undeformed volume to natural coordinates $\boldsymbol{\xi}_j$. To do so, we subtract the "origin" of the element (vertex closest to the origin for a hexahedron with positive coordinates) from the quadrature point, then scale the resulting vector with the inverse of the side lengths of the hexahedron.

5.2 Cut Elements

To integrate the strain energy for elements cut into subvolumes, the extended finite element method introduces additional "enriched" degrees of freedom together with specifically constructed shape functions. Our enrichment strategy is mathematically equivalent to Shifted Sign Enrichment (SSE), which was recently used by Koschier et al. [2017] to *explicitly* represent strong discontinuities along detailed, piecewise linear cuts. We will first discuss standard SSE, and then present a reformulation that reduces the computational overhead for (multi-)enriched elements and simplifies implementation.

For elements cut into multiple subvolumes V_j, standard SSE adds enriched shape functions to ensure that the function space is *complete* on *every* subvolume V_j

$$\mathbf{x}(\boldsymbol{\xi}) = \sum_i \left(N_i(\boldsymbol{\xi}) \mathbf{x}_i + \sum_{j : V_j \neq V_i} \mathbf{1}_{V_j}(\boldsymbol{\xi}) N_i(\boldsymbol{\xi}) \mathbf{x}_i^j \right) \tag{19}$$

where $\mathbf{x}_i^j$ are the additional degrees of freedom and $\mathbf{1}_{V_j}$ is the characteristic function that evaluates to one if a point with natural coordinates $\boldsymbol{\xi}$ is contained in V_j and zero otherwise. To ensure completeness, the second sum runs over all subvolumes except the one we "assign" the original shape function N_i to, denoted by V_i. Refer to Fig. 9 left for a 1D example where two linear basis functions are enriched to ensure that the function space is linear on all three "subvolumes". The formulation used by Koschier et al. is identical to this one except for signs.

Replacing the interpolation of deformed nodes in the deformation gradient with this enriched interpolation, and integrating the strain energy density over the individual subvolumes V_j, we can accurately simulate the elastic response of the body enclosed in a B-rep. However, the traditional approach (Eq. 19) of *adding* enriched basis functions to the existing set has practical shortcomings: For enriched elements, the elastic energy depends on the additional degrees of freedom, meaning that the elemental force vector and tangent stiffness matrix grow in size. For example, for linear hexahedral elements cut into k subvolumes, the energy gradient has size $24k$, and the energy Hessian size $24k \times 24k$. Due to the dependence of the elemental energy and energy derivative evaluations on the *number* of subvolumes, custom code is required for every discrete number of subvolumes.

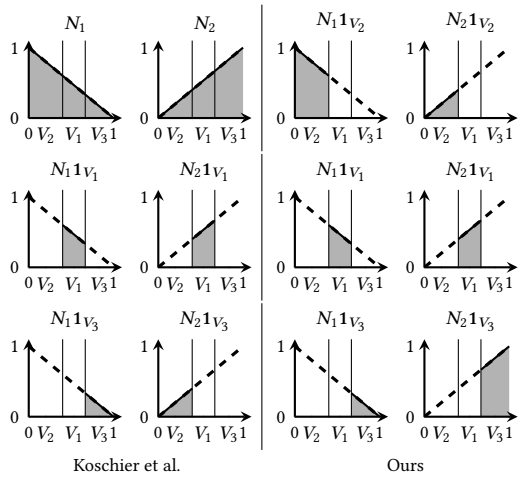


Fig. 9. **Change of Basis** Comparing our (right) to Koschier et al.'s basis (left) for a 1D example where the domain is partitioned into "subvolumes" V_1, V_2 and V_3, we observe that both are trilinear on all three subvolumes while the support of the individual basis functions is different: while V_2 and V_3 are affected by three, and V_1 by four DOFs for Koschier et al.'s basis, all subvolumes are only affected by two DOFs for our basis.

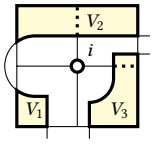
To resolve these practical roadblocks, we propose to perform a change of basis

$$\mathbf{x}(\boldsymbol{\xi}) = \sum_j \left(\sum_i \mathbf{1}_{V_j}(\boldsymbol{\xi}) N_i(\boldsymbol{\xi}) \mathbf{x}_i^j \right). \tag{20}$$

As illustrated in Fig. 9 right (1D example), this basis spans the same function space, but the support of the basis functions for each subvolume are *local* to their domain. This reformulation has far-reaching benefits. Because we can treat each subvolume V_j of an enriched element like a standard element with nodal degrees of freedom $\mathbf{x}_i^j$ for every N_i, a standard FEM implementation can be used to evaluate the elemental elastic energy and its derivatives. Moreover, unlike the traditional formulation of strong-discontinuity enrichments (Eq. 19), the computational complexity of Hessian evaluations scales *linearly* instead of *quadratically* with the number of subvolumes. For example, for linear hexahedral elements, only k evaluations of 24×24 Hessian matrices are needed (instead of one $24k \times 24k$ matrix). Because intersections of CAD models with hexahedral meshes tend to cut elements into large sets of subvolumes, we observe a remarkable increase in simulation and optimization performance. Additionally, while the enriched basis in Eq. 19 does *not* fulfill the partition of

unity property, our enriched basis does. This guarantees that rigid body movement can be correctly represented even if cut elements are present [Liu 2016].

It remains to discuss which degrees of freedom $\mathbf{x}_i^j$ are shared between incident elements, and which ones are kept separate: For every element incident to vertex i, we check if the elemental subvolumes are connecting through grid faces. As we illustrate in the inset in 2D, we consider connected subvolumes (volumes separated by dotted lines) to be one entity V_j, adding internal force and tangent stiffness contributions to a *shared* degree of freedom $\mathbf{x}_i^j$. We therefore have three degrees of freedom $\mathbf{x}_i^1$, $\mathbf{x}_i^2$, $\mathbf{x}_i^3$ instead of five in the inset example. For integration, we consider elemental subvolumes separate entities.



5.3 Boundary Conditions and Gravity

Unlike conformal meshes, our nodal degrees of freedom do not lie on the surface, and we cannot enforce Dirichlet conditions by holding a subset of them fixed. We instead rely on Nitsche's method [1971], a technique well-known in mechanical engineering.

To enforce prescribed displacements $\hat{\mathbf{u}}(\boldsymbol{\xi})$ on a part of the boundary S_{disp}, we introduce a *displacement* energy

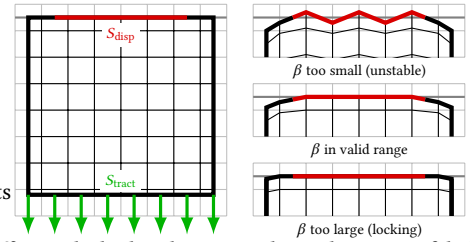
$$E_{\text{disp}}(\mathbf{x}, \boldsymbol{\lambda}) = \int_{S_{\text{disp}}} \boldsymbol{\lambda}(\boldsymbol{\xi}) \cdot (\mathbf{u}(\boldsymbol{\xi}) - \hat{\mathbf{u}}(\boldsymbol{\xi}))\, \mathrm{d}S, \tag{21}$$

setting $\mathbf{u}(\boldsymbol{\xi}) = \mathbf{x}(\boldsymbol{\xi}) - \mathbf{X}(\boldsymbol{\xi})$. The Lagrange multiplier *function* $\boldsymbol{\lambda}(\boldsymbol{\xi})$ can be thought of as *reaction traction* acting on the surface S_{disp} to enforce the prescribed displacements. Nitsche showed that $\boldsymbol{\lambda}$ equals $-\mathbf{P} \cdot \mathbf{n}$ where $\mathbf{P} = \frac{\partial \Psi}{\partial \mathbf{F}}$ is the first Piola-Kirchhoff stress (PK1) and $\mathbf{n}$ the surface normal evaluated at $\boldsymbol{\xi}$. This substitution removes $\boldsymbol{\lambda}$ as an unknown variable.

In a discrete setting, this formulation is unstable, and we add the common stabilizer

$$\frac{\beta}{2} \int_{S_{\text{disp}}} \|\mathbf{u}(\boldsymbol{\xi}) - \hat{\mathbf{u}}(\boldsymbol{\xi})\|^2 \, \mathrm{d}S \tag{22}$$

with stabilization parameter β, to E_{disp}. If β is chosen too small, the method remains unstable. If it is too large, elements intersecting the boundary S_{disp} effectively lock. Chosen in the right range, β balances the enforcement of prescribed displacements with the elastic response of the model as illustrated in the inset.



Surface tractions $\hat{\mathbf{t}}(\boldsymbol{\xi}) : \mathbb{R}^3 \to \mathbb{R}^3$ that act on another part S_{trac} of the boundary (see above inset) can be added as in standard FEM with a *traction* energy

$$E_{\text{trac}}(\mathbf{x}) = \int_{S_{\text{trac}}} \hat{\mathbf{t}}(\boldsymbol{\xi}) \cdot \mathbf{u}(\boldsymbol{\xi})\, \mathrm{d}S. \tag{23}$$

Note that for integration over the domains S_{disp} and S_{trac}, our rules for curved surface domains can readily be used.

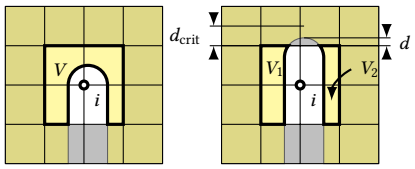


Fig. 10. **Differentiability** A concave feature of the model (yellow) intersects the hexahedral mesh adjacent to a simulation node i in a single connected component V (left). If the feature moves past the boundary of the adjacent elements, the intersection splits into two parts V_1 and V_2 (right), and the simulation node into two enriched degrees of freedom.

Minimizing the total potential energy E with $E_{\text{ext}} = E_{\text{trac}} + E_{\text{disp}}$, we can solve for the equilibrium state $\mathbf{x}$, while accounting for features at subelement resolution. For models where gravity is non-negligible, we add the energy $E_{\text{gravity}}(\mathbf{x}) = \int_V \rho\, \mathbf{g} \cdot \mathbf{u}(\boldsymbol{\xi})\, \mathrm{d}V$ with density ρ to the external energy. The constant 3D vector $\mathbf{g}$ points in the direction of gravity and has magnitude equal to the gravitational acceleration.

5.4 Differentiability

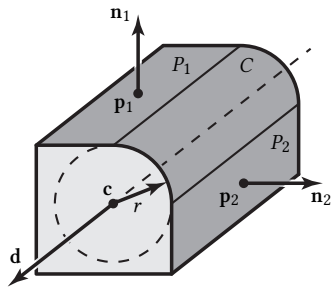
In its current form, our XFEM formulation is sufficiently smooth and differentiable with one exception as we illustrate in Fig. 10 with an example in 2D: if shape parameters change, a subvolume V could split into two subvolumes V_1 and V_2 in the 1-neighborhood of a vertex i. As a consequence, the degree of freedom $\mathbf{x}_i$ is split into two enriched degrees of freedom $\mathbf{x}_i^1$ and $\mathbf{x}_i^2$, and the elastic response of the model may become discontinuous as a function of the shape parameters. Note that our simulation is only discontinuous at the point of the split.

To detect cases that could lead to a discontinuity, we compare the subvolumes in the 1- and 2-neighborhood of a vertex. If two (or more) subvolumes in the 1-neighborhood are connected in the 2-neighborhood, adjustments to shape parameters could lead to a discontinuity.

To avoid these discontinuities, we define a metric d that is zero at the point where a volume splits, and varies in the range $[0, d_{\text{crit}}]$ if the two subvolumes are close to merging. A metric that lends itself is the shortest orthogonal distance between the boundary of the 1-neighborhood of vertex i and a geometric saddle point in the model (compare with Fig. 10 right). To ensure that degrees of freedom smoothly merge as d goes to zero, we add penalties of the form

$$b(d)\|\mathbf{x}_i^1 - \mathbf{x}_i^2\|^2 \quad \text{with} \quad b(d) = \log^2\left(\min\left\{1, \frac{d}{d_{\text{crit}}}\right\}\right) \tag{24}$$

to our potential energy E. This procedure guarantees a smooth transition of the simulation result as concave features pass through the simulation mesh. For numerical reasons, we cut off b at a high value, so the conditioning of the discretized PDE does not deteriorate. Note that we have not observed any negative effects due to this choice.



$$\text{Constraints}(P_1, C):$$
$$(\mathbf{p}_1 - \mathbf{c}) \cdot \mathbf{n}_1 - r = 0$$
$$\mathbf{d} \cdot \mathbf{n}_1 = 0$$
$$\text{Constraints}(P_2, C):$$
$$(\mathbf{p}_2 - \mathbf{c}) \cdot \mathbf{n}_2 - r = 0$$
$$\mathbf{d} \cdot \mathbf{n}_2 = 0$$

Fig. 11. **Tangential Constraints** A cylindrical segment C is tangent to two adjacent planes P_1 and P_2. To maintain the tangencies, constraint equations are generated and added to $\mathbf{c}_{\text{para}}$. The surface parameters $\mathbf{p}_1, \mathbf{n}_1, \mathbf{p}_2, \mathbf{n}_1, \mathbf{c}, r, \mathbf{d}$ are part of the parameter vector $\mathbf{p}$.

6 OPTIMIZING SHAPE PARAMETERS

Our shape optimization enables a wide range of applications, including combined mass distribution and strength-to-weight ratio, rest shape optimization, and various other inverse design problems that require an accurate integration of properties or discretized PDE equations over the parameterized design domain enclosed by a B-rep.

A key advantage of our approach is that our hexahedral simulation mesh is *independent* of our parameterized boundary representation. Hence, in the context of strength-to-weight ratio or rest shape optimization, we can work with the *same* hexahedral mesh, even under large changes of shape parameters.

Before we discuss how to efficiently compute derivatives of integration rules, the elastic response, and the user-specified parameterization, we will refine our parameterization formulation.

6.1 Parameterization

A good parameterization should provide sufficient degrees of freedom to enable meaningful improvements of combinations of objectives, while preserving the original design intent of the CAD model. To this end, shape parameters should describe its shape in an intuitive manner, enabling the user to easily select a subset of them for optimization.

On a low level, we represent CAD models as a set of NURBS patches as described in Sec. 3. We collect all NURBS control points in a vector $\mathbf{q}$. CAD models typically contain many geometric primitives, such as, e.g., planar, cylindrical, or toroidal segments. These are more intuitively described using mid-level parameters such as, e.g., the central axis and radius of a cylinder. We collect all these parameters in a vector $\mathbf{p}$. Control points in $\mathbf{q}$ necessarily depend on the values of parameters in $\mathbf{p}$, e.g., changing the radius of a cylinder moves the NURBS control points describing the cylinder's geometry. We denote this mapping as $\mathbf{q} = \mathbf{q}(\mathbf{p})$. By only modifying $\mathbf{q}$ indirectly through $\mathbf{p}$, primitives retain their basic shape.

However, additional constraints are required to preserve the design intent of a CAD model. A frequent example is a fillet, i.e., an initially sharp edge that has been rounded off by adding a cylinder segment, as seen in Fig. 11. The cylinder C is tangent to the planes P_1 and P_2, and this tangency must be preserved when modifying $\mathbf{p}$; otherwise, unintended sharp edges may be introduced. Constraints of this type can be formalized as equations in $\mathbf{p}$, as exemplified for the fillet in Fig. 11.

Once a CAD model is loaded, we traverse all pairs of adjacent surfaces and detect tangency relationships that need to be preserved. This is done automatically using a look-up table that stores possible relationships between primitive types, such as the one between a cylinder and a planes in the fillet example above. Whenever a situation like this is encountered, a list of implicit constraints is generated for the pair of surfaces in question. Processing the entire model in this manner yields a non-linear constraint system $\mathbf{c}_{\text{para}}(\mathbf{p}, \mathbf{q}(\mathbf{p})) = \mathbf{0}$ that needs to remain satisfied throughout optimization. In addition to these automatic constraints, our UI enables the user to define high-level model parameters by grouping specific parameters in $\mathbf{p}$, to keep certain surfaces fixed, or to enforce symmetries in the model. These additional constraints are also added to $\mathbf{c}_{\text{para}}$.

During optimization, it may occur that no value of $\mathbf{p}$ exists which satisfies the tangency and user-specified constraints in $\mathbf{c}_{\text{para}}$ at the same time. To ensure that the CAD model remains valid regardless, we introduce *effective* shape parameters $\mathbf{p}_\perp$, which represent the projection of the given parameters $\mathbf{p}$ onto the constraint manifold defined by $\mathbf{c}_{\text{para}}$. This projection together with $\mathbf{q} = \mathbf{q}(\mathbf{p}_\perp)$ defines an *implicit* mapping from model parameters to NURBS control points. To summarize, we minimize

$$f_{\text{para}}(\mathbf{p}_\perp, \mathbf{q}) = \frac{1}{2}\|\mathbf{p}_\perp - \mathbf{p}\|^2 + \frac{1}{2}\|\mathbf{q}(\mathbf{p}_\perp) - \hat{\mathbf{q}}\|^2 \tag{25}$$

over the constraint manifold spanned by $\mathbf{c}_{\text{para}}$, or the corresponding Lagrangian

$$\mathcal{L}(\mathbf{p}_\perp, \mathbf{q}, \boldsymbol{\lambda}) = f_{\text{para}}(\mathbf{p}_\perp, \mathbf{q}) - \boldsymbol{\lambda}^T \mathbf{c}_{\text{para}}(\mathbf{p}_\perp, \mathbf{q}(\mathbf{p}_\perp)) \tag{26}$$

to first-order optimality. Here, $\mathbf{p}$ is the set of shape parameters modified through an optimization step, and $\mathbf{p}_\perp$ is its projection onto $\mathbf{c}_{\text{para}}$. Insufficiently constrained control points $\mathbf{q}(\mathbf{p}_\perp)$ are modified as little as possible by keeping them close to their initial values $\hat{\mathbf{q}}$.

6.2 Shape Optimization and Derivatives

After the user specifies a desired parameterization together with a set of objectives that depend on the parameterization and the elastic response of the model, we aim at solving the first-optimality constrained problem

$$\min_{\mathbf{p}} f(\mathbf{p}_\perp(\mathbf{p}), \mathbf{q}(\mathbf{p}_\perp), \mathbf{x}(\mathbf{p}_\perp)) \text{ s.t. } \begin{bmatrix} \mathcal{L}_{\mathbf{p}_\perp} \\ \mathcal{L}_{\mathbf{q}} \\ \mathcal{L}_{\boldsymbol{\lambda}} \end{bmatrix} = 0 \text{ and } E_{\mathbf{x}} = 0. \tag{27}$$

Note that, compared to our formulation as outlined in our overview (Sec. 3), we replace the direct dependence of the objective on the parameters with an implicit dependence $\mathbf{p}_\perp(\mathbf{p})$. Posing a design optimization in this particular way is advantageous because, during numerical optimization, the parameters $\mathbf{p}$ can take on values that do not fulfill our parameterization constraints, for example during line search along a descent direction. The combination of a continuous "projection" $\mathbf{p}_\perp(\mathbf{p})$, formulated with a first-order optimality constraint on a parameterization Lagrangian, and the use of only valid sets of parameters in objective evaluations, enables the use of a standard quasi-Newton for optimization where first-order optimality constraints are *implicitly* enforced.

In objective and objective gradient evaluations for a particular $\mathbf{p}$, we first solve the Lagrangian to first-order optimality. We then use the resulting set of control points $\mathbf{q}(\mathbf{p}_\perp)$ in the minimization of the potential energy E to equilibrium, $E_{\mathbf{x}}(\mathbf{q}(\mathbf{p}), \mathbf{x}(\mathbf{p})) = 0$. To compute shape derivatives of our objective with respect to shape parameters

$$\mathrm{d}_{\mathbf{p}} f = f_{\mathbf{p}_\perp} \mathrm{d}_{\mathbf{p}} \mathbf{p}_\perp + f_{\mathbf{q}} \mathrm{d}_{\mathbf{p}} \mathbf{q} + f_{\mathbf{x}} \mathrm{d}_{\mathbf{p}} \mathbf{x}, \tag{28}$$

we apply the implicit function theorem to our parameterization

$$\begin{bmatrix} \mathcal{L}_{\mathbf{p}_\perp,\mathbf{p}_\perp} & \mathcal{L}_{\mathbf{p}_\perp,\mathbf{q}} & \mathcal{L}_{\mathbf{p}_\perp,\boldsymbol{\lambda}} \\ \mathcal{L}_{\mathbf{q},\mathbf{p}_\perp} & \mathcal{L}_{\mathbf{q},\mathbf{q}} & \mathcal{L}_{\mathbf{q},\boldsymbol{\lambda}} \\ \mathcal{L}_{\boldsymbol{\lambda},\mathbf{p}_\perp} & \mathcal{L}_{\boldsymbol{\lambda},\mathbf{q}} & \end{bmatrix} \begin{bmatrix} \mathrm{d}_{\mathbf{p}} \mathbf{p}_\perp \\ \mathrm{d}_{\mathbf{p}} \mathbf{q} \\ \mathrm{d}_{\mathbf{p}} \boldsymbol{\lambda} \end{bmatrix} = - \begin{bmatrix} \mathcal{L}_{\mathbf{p}_\perp,\mathbf{p}} \\ \mathcal{L}_{\mathbf{q},\mathbf{p}} \\ \mathcal{L}_{\boldsymbol{\lambda},\mathbf{p}} \end{bmatrix} \tag{29}$$

and quasi-static equilibrium $E_{\mathbf{x},\mathbf{x}} \mathrm{d}_{\mathbf{p}} \mathbf{x} = -E_{\mathbf{x},\mathbf{q}} \mathrm{d}_{\mathbf{p}} \mathbf{q}$ where we use $(.)_{\mathbf{p}}$ for partial, and $\mathrm{d}_{\mathbf{p}}(.)$ for total derivatives. For efficiency, we rely on the adjoint method.

6.3 Taking Derivatives of Quadrature Rules

If we make adjustments to our shape parameters, the volume V changes, and hence the domains of our hierarchical rules. While these changes are restricted to elements that were or are newly cut by the boundary, the cost of taking derivatives of rules can be considerable and requires a significant amount of bookkeeping. To reduce the computational complexity and simplify the implementation of shape derivatives, we describe how we can avoid some of the terms in our volume, area, and surface rules. Note that rule construction only depends on the control points of the B-rep, hence we can safely ignore other dependencies here.

Area, Surface, and Volume Rules. To keep the matrix $\mathbf{A}$ in the moment fitting equation (Eq. 11) constant, we propose to transform the non-standard domains. However, the right-hand side $\mathbf{b}$ depends on the shape of the non-standard domain, hence the quadrature points and weights, in general, depend on the shape parameters

$$\frac{\mathrm{d}}{\mathrm{d}\mathbf{p}} \int_{D(\mathbf{q})} g(\mathbf{X}) \, \mathrm{d}D = \sum_j \left(\frac{\mathrm{d}w_j}{\mathrm{d}\mathbf{p}} g(\mathbf{X}_j) + w_j \frac{\partial g(\mathbf{X})}{\partial \mathbf{X}} \frac{\mathrm{d}\mathbf{X}_j}{\mathrm{d}\mathbf{p}} \right). \tag{30}$$

A key observation to increase the efficiency of rule derivatives, is, that, if we keep the transformations for volume, area, and surface integrals *after initial* rule construction fixed, the quadrature points *no longer* depend on the shape parameters for volume, area, and surface rules

$$\frac{\mathrm{d}}{\mathrm{d}\mathbf{p}} \int_{D(\mathbf{q})} g(\mathbf{X}) \, \mathrm{d}D = \sum_j \frac{\partial w_j}{\partial \mathbf{p}} g(\mathbf{X}_j) \quad \text{for} \quad D \in \{A, S, V\}. \tag{31}$$

Note that for edge and curve rules, the transformation from the general domain $[a, b]$ to a standard domain $[0, 1]$ is necessary. Otherwise, we cannot apply tabulated Gauss-Legendre rules. However, if we keep applying the initial transformation for area, surface, and volume rules, the weights, computed with the moment fitting equation, only depend on the right-hand side $\mathbf{b}$ but not on a shape-dependent transformation. Crucially, these shape derivatives are *exact* if function g is in the function space spanned by the bases used for rule construction.

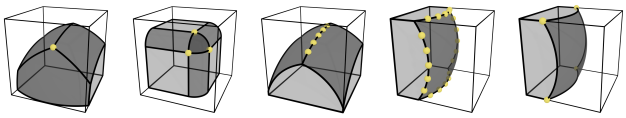


Fig. 12. **Derivatives of Curve Rules** To parameterize curve integrals, we sample intersection curves between patches and hexahedral grid planes. We differentiate the following cases (from left to right): intersection points between three or more model surfaces (cases 1 and 2), sample points on an intersection curve between two surfaces (case 3), sample points on surface-grid intersection curves (case 4), and intersection vertices between a surface and two grid planes (case 5).

Curve Rules. As we pointed out in Sec. 4, it is, in general, not possible to extract an analytical parameterization for intersection curves that arise when several NURBS patches intersect within a hexahedral element, or a NURBS intersects with one of the element planes. Hence, we represent these planar or spatial curves with sample points, and differentiate between several cases, illustrated in Fig. 12.

During optimizations, we make changes to shape parameters, and implicitly also to control points. Changes to the control points, in turn, move the position of sample points on intersection curves. To treat NURBS patches and element planes the same, we parameterize the latter, defining a mapping from parameter values $\mathbf{u} = [u, v]^T$ to plane points $\hat{\boldsymbol{\sigma}}(\mathbf{u}) \in \mathbb{R}^3$. A sample point on two or more "surfaces" is then defined by a pair of uv-coordinates for each surface. To be able to take shape derivatives, it is important to understand the relationship between these coordinates and the shape parameters.

To this end, it is best to look at a specific example (case 1 or 5 in Fig. 12) where three "surfaces" intersect in a single point: if we change $\mathbf{p}$, the control points of the three patches and also the uv-coordinates in their respective parameter domain, change. What uniquely defines the uv-coordinates is the constraint that they all map to the *same* point in 3D, formalized with an equation $\hat{\boldsymbol{\sigma}}_i(\mathbf{u}_i(\mathbf{p}), \mathbf{q}(\mathbf{p})) - \hat{\boldsymbol{\sigma}}_j(\mathbf{u}_j(\mathbf{p}), \mathbf{q}(\mathbf{p}))$ for every pair (i, j) of "surfaces". Collecting these equations in a system $\Sigma = 0$, and the uv-coordinates in a vector $\mathbf{U}$, we apply the implicit function theorem to compute analytical derivatives $\mathrm{d}_{\mathbf{p}} \mathbf{U} = -\Sigma_{\mathbf{U}}^{-1} \Sigma_{\mathbf{q}} \, \mathrm{d}_{\mathbf{p}} \mathbf{q}$.

However, this particular case is rare because real-world CAD models typically have filleted edges and corners with either all adjacent surfaces or subsets of them being tangent (case 2). In such cases, the Jacobian $\Sigma_{\mathbf{U}}$ becomes rank-deficient. Another case that leads to a rank-deficiency in $\Sigma_{\mathbf{U}}$ arises if we place sample points on intersection curves that are defined by *two* "surfaces" (case 3 or 4 in Fig. 12).

To be able to compute derivatives in these cases, we first classify sample points by analyzing the normals of adjacent surfaces. We then complement the equations in $\Sigma = 0$ with planes that span the null space. For example, for a sample point on a sharp edge (cases 3 and 4), we define a plane whose normal is set to the cross product of the two surface normals. Because the components of derivatives that lie in this null space do not change the value of our integrals to first order, we can safely ignore them after computing the derivatives.

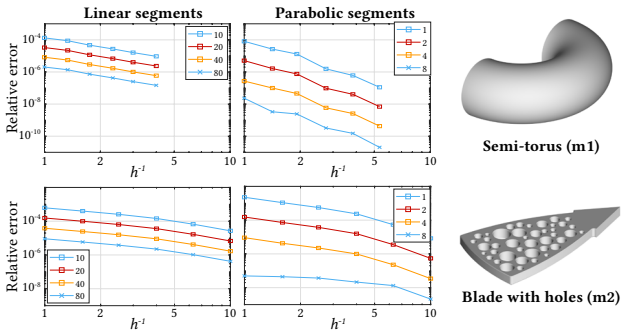


Fig. 13. **Quadrature convergence.** Relative error of volume integrals under refinement of integration grid and refinement of curve integral approximation. The cell side length in the integration grid is given by h, and is isotropically refined in all three directions. Each curve (cf. Sec. 4.1) is split into a constant number of segments, which is given in the legend. The graphs compare a piecewise-linear curve approximation and a piecewise-parabolic curve approximation. Convergence is faster for parabolic segments, and accuracy improves with both types of refinement.

7 RESULTS

Accuracy of Quadrature. We evaluated the accuracy of our quadrature scheme by comparing the results to a ground truth and to the results obtained with Müller et al.'s method [2013]. For evaluation, we intersected the model shown in Fig. 4 with a regular grid to yield a large number of geometric subvolumes, and selected a set of 23 test monomials. The ground truth was obtained by finely triangulating every subvolume and analytically integrating every monomial.

We generated one integration rule for each subvolume using our method and one using Müller et al.'s method. For a fair comparison, we used the same quadrature point locations on surfaces and in volumes, and the same polynomial basis. As an error metric, we use the absolute difference between the ground truth integral and the quadrature solution, divided by the integral over the surrounding element. All combinations of subvolumes and monomials yield 18,078 data points in total.

Fig. 4 shows that our method yields a worst-case relative error of 8.6×10^{-4}, with the vast majority of samples below 10^{-4}. Müller et al.'s method may fail to produce usable rules on curved surfaces if the volume contains few planar surfaces (compare with Fig. 6), or if the distribution of quadrature points on curved surfaces is unfavorable.

Convergence of Quadrature. Fig. 13 shows additional tests for the convergence of integration results. We performed volume integral tests on an analytical surface (a semi-torus, m1), and a model with small topological features (m2). Three parameters were varied during the test: the tessellation level h of the regular grid, the number of segments per curve used to approximate line integrals (cf. Sec. 4.1), and the order of these segments (linear or parabolic).

At the lowest tessellation level, the models are embedded in grids of $6 \times 12 \times 4$ cells for m1 and $10 \times 1 \times 6$ cells for m2, respectively. For m2, this is sufficiently coarse that individual cells may contain several holes. Relative integration errors of down to 10^{-7} may still be reached by increasing the number of segments per curve. The

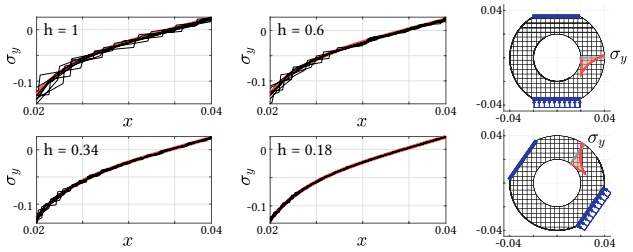


Fig. 14. **Stress convergence.** Simulation of ring in compression using XFEM, embedded in the simulation grid in different orientations (right). Vertical stresses along horizontal cross-section (left) for different h-refinement levels and different rotations (gray curves), and ground truth (red curve).

comparison between segment types shows that parabolic segments are more economical; two parabolic segments achieve approximately the same accuracy as twenty linear segments. The diagrams also indicate faster convergence rates for parabolic segments, which matches theoretical predictions [Atkinson and Venturino 1993]. The computational cost per segment is similar for linear and parabolic segments, as they use the same number of integration points.

Comparison to Standard FEM. We performed a simulation test of our XFEM formulation on a standard example of a ring in compression. This was to verify that the extended formulation with Nitsche's method on Dirichlet conditions converges to the same result as standard FEM. A front view of the ring model is shown in Fig. 14 (top, right). The blue bars denote Dirichlet conditions, with an enforced compressive displacement on the bottom. Vertical stress, σ_y, along a horizontal cross section is shown as a red curve.

To ensure stability under different embeddings, the simulation was repeated for eight different rotations of the model within the grid. This leads to Dirichlet conditions and stress measurements at different inclinations, as indicated in Fig. 14 (bottom, right). The four plots on the left show XFEM stresses for different h-refinement levels, where $h = 1$ corresponds to the tessellation shown in the figure. The ground truth obtained from high-resolution standard FEM is shown as the red curve, and the measurements from differently rotated XFEM meshes are shown as gray curves. The plots show that the stresses converge consistently for all rotations.

Aircraft. Our differentiable simulator can deal with models of high geometric complexity, exemplified by a model of the internal structure of an aircraft where we simulate the deflection of the wing under the load of the aircraft (Fig. 1 left).

7.1 Shape Optimization

In the remainder of this section, we will discuss our optimization results. To visualize parameterizations, we use yellow for parameterized surfaces and orange for surfaces that move due to tangent preservation constraints. Dirichlet conditions are shown in red and surface tractions in green.

Wrench. The wrench model (see Figs. 2 and 17 top row) is composed of a set of NURBS patches that are subject to many tangency relationships along its edges. The model is parameterized by the

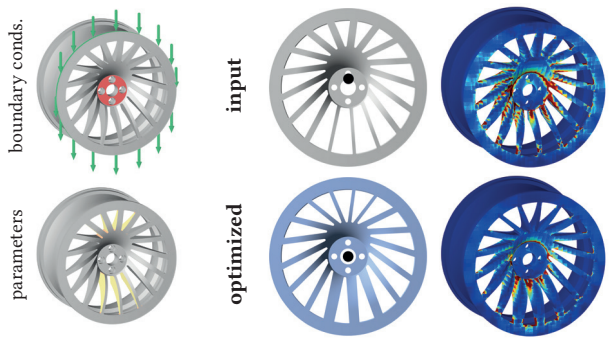


Fig. 15. **Asymmetric Wheel.**

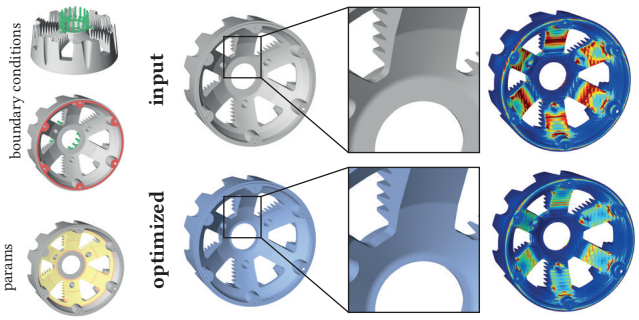


Fig. 16. **Motor Housing.**

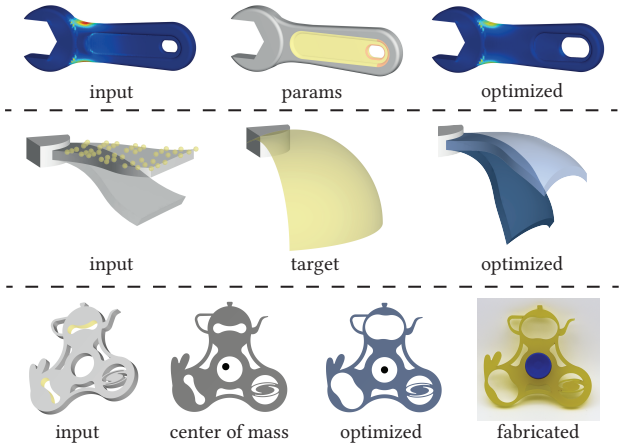


Fig. 17. **Wrench, Lampshade, and Fidget Spinner.**

height and width of the slotted indentation in the handle, the height and width of the hole in the handle, and the fillet radius around the hole. Assigning a linear elastic material and parameters for a standard tool steel, we optimize the strength-to-weight ratio under a mechanical advantage setting where a torque is applied to a nut.

Our optimization co-optimizes a type one objective (Eq. 4), whose integrand is set to the exponentiated distance of the Cauchy stress to the von Mises failure surface [Schumacher et al. 2018], and a type two objective (Eq. 5) integrating the model's density. As we can see in our supplemental video and in Fig. 17, our shape optimization succeeds in significantly improving the mode's strength, with the handle hole geometry smoothly passing through several hexahedral elements. Note that our parameterization preserves all tangencies.

Asymmetric Wheel and Fidget Spinner. While many different designs for automobile wheels exist, they are typically rotationally symmetric in order to ensure that they will spin stably around their axis and not introduce vibrations. A combination of a type one (Eq. 4 with integrand set to Ψ for linear elasticity) and a set of type two objectives (Eq. 5; integrals over all monomials) enables the co-optimization of the strength-to-weight ratio and mass distribution of an *asymmetric* wheel design (see Fig. 1 middle and Fig. 15). As we demonstrate in the accompanying video with scaled-down, 3D printed wheels, the unoptimized input wobbles while our optimized design spins stably. The reason for this difference is that the center of mass (black spheres in the second column in Fig. 15) does not lie on the spinning axis (9.9 mm off for a wheel of 40 cm diameter; 22 mm off when considering the mass properties of the wheel spokes and hub only) for the unoptimized design. The parameterization used in the optimization allows for the width of the spokes of the wheel to vary, and the additional compliance term ensures that the result is not only functional (CoM misalignment <0.1 mm) but also optimal from a structural point of view (see stress visualizations in Fig. 15). While structural considerations are less relevant in toy design, the use of only type two objectives enables the design of asymmetric spinning toys such as a fidget spinner that combines (with some imagination) a teapot, a SIGGRAPH logo, and a bunny in a unified design (see Fig. 17 bottom row).

Motor Housing. In this example, we optimize the compliance of an aluminium housing for an electric motor, with cooling fins on the outside. Tailored for manufacturing by casting or machining, it is of paramount importance to keep the model undercut-free. Moreover, to preserve its function and appearance, it is essential to enforce rotationally symmetric changes. To this end, we parameterize the thickness and width of the 6 "spokes" with a total of 4 parameter (Fig. 16 params), constraining the normals of the surfaces that are orthogonal to the symmetry axis in the initial design, to remain orthogonal throughout optimizations. Relying on a type one objective with Ψ set to linear elasticity, we are able to reduce the overall compliance of the model by 6.7%.

Lampshade. Targeting furniture design, we perform hyperelastic rest shape optimization [Chen et al. 2014] on an initially flat lampshade design that consists of six rotationally symmetric, curved blades protruding from the central unit containing the light fitting (see Fig. 1 right and Fig. 17 middle row). In contrast to our other examples, we parameterize the *continuous* outer surface of the lampshade design (a total of 48 control points) and optimize the model's rest shape (type three objective, Eq. 6) such that the outer surface deforms into a toroidal target shape under self-weight. During optimizations, we keep the curvy silhouette fixed, and constrain the inner surface to move with the outer surface with a thickness preservation constraint. To manufacture the optimized lampshade, we use MoldStar 30 rubber. As can be seen in our accompanying

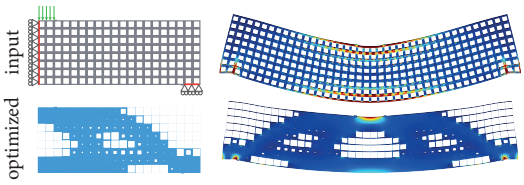


Fig. 18. **Topology Optimization.**

video and in Fig. 17, the lampshade deforms into the desired target shape (max. target matching error is 16 mm before and 0.6 mm after optimization). Because Schulz et al.'s technique [2017] does not scale well with the number of parameters, shape optimizations like this one would be difficult to perform with their method. For accurate simulations, we rely on a Mooney-Rivlin model.

Topology Optimization. Although our framework does not currently support topology changes, we can mimic topology optimization by optimizing a grid of 20×8 square holes (total of $n = 160$ shape parameters). The objective is to design a symmetric bridge that minimizes compliance under a volume constraint. The initial grid with boundary conditions, the optimized grid, and the deformed states are shown in Fig. 18.

The side length l_i of any square hole is allowed to vary within $0 < l_i < 1$. The true volume fraction of a geometric cell is given by $V_i = 1 - l_i^2$. To drive parameters towards extremal values, we use $\tilde{V}_i = V_i^{2/3}$ to penalize the volume of intermediate cells relative to their stiffness, and constrain $\frac{1}{n}\sum_i \tilde{V}_i = 0.7$. The simulation uses an 80×32 grid, so each cell is meshed by a 4×4 block of elements, some of which appear and disappear as walls move through them. The result is in line with conventional topology optimization results. Moreover, this demonstrates the capability of our method to deal with large numbers of optimization parameters as well as very thin model sections.

Performance. All simulations and optimizations were performed on a single core of an Intel Core i7-8700. Tab. 1 provides data about the complexity of models and timings of optimization and simulation routines. We rely on *linear* shape functions in all our simulations.

Table 1. **Simulation and Optimization Statistics** Columns from left to right: name of the demonstration; resolution of simulation grid; number of integration subvolumes (#SV); number of shape parameters (#P); number of NURBS patches in the CAD model (#S); time per iteration or total simulation time (Aircraft) in seconds (t); number of iterations (i).

Demonstration	Grid	#SV	#P	#S	t	#i
Aircraft	160x40x160	15,840	-	679	104	-
Wrench	32x4x12	926	5	54	8.0	64
Fidget Spinner	16x16x1	161	12	82	3.2	28
Asym. Wheel	10x20x20	1,652	14	154	70	31
Motor Housing	30x30x18	6,628	4	557	86	43
Lampshade	26x26x38	1,954	48	6	352	17
Topology Opt.	80x1x32	2,143	160	653	50	33

8 CONCLUSION

We have devised a generic shape optimization that enables the solution of a wide range of computational design problems directly on a CAD representation. While we have developed our hierarchical integration and extended finite element formulation with an eye on shape optimization, we see applications beyond the discussed context. For example, because our simulation of CAD models is fully *differentiable*, it is well suited for applications in geometric deep learning [Koch et al. 2019].

Our proposed change of basis of enriched shape functions makes it straightforward to turn an existing FEM implementation into one that supports strong discontinuities in cut elements. Moreover, it makes Hessian computations for cut elements significantly more efficient, and leads to a basis that preserves not only the Kronecker delta but also the partition of unity property. A change of basis as used herein would be beneficial in XFEM applications in general.

Furthermore, we introduced a parameterization Lagrangian that enables optimization with a set of parameters that does not lie on the constraint manifold defined by a user-specified set of constraints. *Implicitly* enforcing the first-order optimality of this Lagrangian in shape optimizations, we can compute analytical gradients of the *continuous* projection of the parameters onto the closest set of valid values. We see utility of this projection in other computational design problems and beyond.

Limitations & Future Work. While our integration schemes and XFEM formulation supports the use of relatively coarse hexahedral meshes, Dirichlet conditions cannot be fulfilled *exactly* along the boundary of the domain. Although we have not experienced any difficulties in choosing a stabilization parameter that prevents locking or instabilities, striking for the right balance between fulfillment of displacement constraints and elastic response may be challenging. If the fulfillment of constraints is unsatisfactory, an increase in resolution, hence a decrease of computational efficiency, is unavoidable. While we rely on an *axis-aligned* hexahedral grid for increased performance, the support of conforming elements in cells that intersect the boundary of a constrained domain, or an adaptive refinement as common in Finite Cell (FC) methods are valid alternatives left for future work.

Another exciting future avenue is the extension of our strong discontinuity formulate to weak discontinuities, enabling applications in dual or multimaterial modeling [Vidimče et al. 2013]. For example, Nitsche's method could be used to treat discontinuities in the deformation gradient.

Finally, our shape optimization does not handle topological changes in the relationships between neighboring NURBS patches. Enforcing their preservation with constraints, we demonstrate with our results that large changes to shape parameters are possible. While it is often desirable to preserve these relationships, there are applications where topological changes are beneficial, constituting an exciting future direction. A differentiated CAD kernel could be useful in this context [Mykhaskiv et al. 2018].

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable feedback; Ilya Baran, and Adriana Schulz and her co-authors [Schulz et al. 2017] for providing CAD models; Dorothea Reusser for help with fabrication. This project has received funding from the European

Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 715767).

REFERENCES

K Atkinson and Ezio Venturino. 1993. Numerical evaluation of line integrals. *SIAM journal on numerical analysis* 30, 3 (1993), 882–888.

Moritz Bächer, Emily Whiting, Bernd Bickel, and Olga Sorkine-Hornung. 2014. Spin-it. *ACM Trans. Graph.* 33, 4 (2014).

Chandrajit L Bajaj, Christoph M Hoffmann, Robert E Lynch, and JEH Hopcroft. 1988. Tracing surface intersections. *Computer aided geometric design* 5, 4 (1988), 285–307.

T. Belytschko and T. Black. 1999. Elastic crack growth in finite elements with minimal remeshing. *Internat. J. Numer. Methods Engrg.* 45, 5 (jun 1999), 601–620.

M. P. Bendsøe and O. Sigmund. 1999. Material interpolation schemes in topology optimization. *Archive of Applied Mechanics (Ingenieur Archiv)* 69, 9-10 (nov 1999), 635–654.

Gaurav Bharaj, David I. W. Levin, James Tompkin, Yun Fei, Hanspeter Pfister, Wojciech Matusik, and Changxi Zheng. 2015. Computational design of metallophone contact sounds. *ACM Trans. Graph.* 34, 6 (2015).

Xiang Chen, Changxi Zheng, Weiwei Xu, and Kun Zhou. 2014. An Asymptotic Numerical Method for Inverse Elastic Shape Design. *ACM Trans. Graph.* 33, 4, Article 95 (July 2014), 11 pages.

J. Austin Cottrell, Thomas J. R Hughes, and Yuri Bazilevs. 2009. *Isogeometric Analysis: Toward Integration of CAD and FEA.* Wiley.

Sachin D. Daxini and Jagdish M. Prajapati. 2017. Parametric shape optimization techniques based on meshless methods: a review. *Structural and Multidisciplinary Optimization* 56, 5 (may 2017), 1197–1214.

Pierre Duysinx, Laurent Van Miegroet, Thibault Jacobs, and Claude Fleury. 2006. Generalized Shape Optimization Using X-FEM and Level Set Methods. In *Solid Mech. and Its App.* Springer Netherlands, 23–32.

François Faure, Benjamin Gilles, Guillaume Bousquet, and Dinesh K. Pai. 2011. Sparse meshless models of complex deformable solids. *ACM Trans. Graph.* 30, 4 (2011), 1.

Thomas-Peter Fries and Ted Belytschko. 2010. The extended/generalized finite element method: An overview of the method and its applications. *Internat. J. Numer. Methods Engrg.* (aug 2010).

G. Haasemann, M. Kästner, S. Prüger, and V. Ulbricht. 2011. Development of a quadratic finite element formulation based on the XFEM and NURBS. *Internat. J. Numer. Methods Engrg.* 86, 4-5 (jan 2011), 598–617.

Ming-Chen Hsu, Chenglong Wang, Austin J. Herrema, Dominik Schillinger, Anindya Ghoshal, and Yuri Bazilevs. 2015. An interactive geometry modeling and parametric design platform for isogeometric analysis. *Computers & Mathematics with Applications* 70, 7 (oct 2015), 1481–1500.

Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 37, 4, Article 60 (July 2018), 14 pages.

Lenka Jeřábková and Torsten Kuhlen. 2009. Stable Cutting of Deformable Objects in Virtual Environments Using XFEM. *IEEE Computer Graphics and Applications* 29, 2 (mar 2009), 61–71.

Peter Kaufmann, Sebastian Martin, Mario Botsch, Eitan Grinspun, and Markus Gross. 2009. Enrichment textures for detailed cutting of shells. *ACM Trans. Graph.* 28, 3 (2009).

Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. 2019. ABC: A Big CAD Model Dataset For Geometric Deep Learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Dan Koschier, Jan Bender, and Nils Thuerey. 2017. Robust eXtended Finite Elements for Complex Cutting of Deformables. *ACM Trans. Graph.* 36, 4, Article 55 (July 2017), 13 pages.

László Kudela, Nils Zander, Tino Bog, Stefan Kollmannsberger, and Ernst Rank. 2015. Efficient and accurate numerical quadrature for immersed boundary methods. *Advanced Modeling and Simulation in Engineering Sciences* 2, 1 (2015), 10.

Grégory Legrain. 2013. A NURBS enhanced extended finite element approach for unfitted CAD analysis. *Computational Mechanics* 52, 4 (apr 2013), 913–929.

G. R. Liu. 2016. On Partitions of Unity Property of Nodal Shape Functions: Rigid-Body-Movement Reproduction and Mass Conservation. *International Journal of Computational Methods* 13, 02 (2016), 1640003.

Haixiang Liu, Yuanming Hu, Bo Zhu, Wojciech Matusik, and Eftychios Sifakis. 2018. Narrow-band Topology Optimization on a Sparsely Populated Grid. *ACM Trans. Graph.* 37, 6, Article 251 (Dec. 2018), 14 pages.

Sebastian Martin, Peter Kaufmann, Mario Botsch, Eitan Grinspun, and Markus Gross. 2010. Unified simulation of elastic rods, shells, and solids. *ACM Trans. Graph.* 29, 4 (2010), 1.

B. Müller, S. Krämer-Eis, F. Kummer, and M. Oberlack. 2017. A high-order discontinuous Galerkin method for compressible flows with immersed boundaries. *Internat. J. Numer. Methods Engrg.* 110, 1 (2017), 3–30.

B Müller, F Kummer, and M Oberlack. 2013. Highly accurate surface and volume integration on implicit domains by means of moment-fitting. *Internat. J. Numer. Methods Engrg.* 96, 8 (2013), 512–528.

M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. 2004. Point based animation of elastic, plastic and melting objects. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation - SCA '04.* ACM Press.

Przemyslaw Musialski, Christian Hafner, Florian Rist, Michael Birsak, Michael Wimmer, and Leif Kobbelt. 2016. Non-linear shape optimization using local subspace projections. *ACM Trans. Graph.* 35, 4 (2016).

Orest Mykhaskiv, Mladen Banovic, Salvatore Auriemma, Pavanakumar Mohanamuraly, Andrea Walther, Herve Legrand, and Jens-Dominik Müller. 2018. NURBS-based and parametric-based shape optimization with differentiated CAD kernel. *Computer-Aided Design and Applications* 15, 6 (2018), 916–926.

Ahmad R. Najafi, Masoud Safdari, Daniel A. Tortorelli, and Philippe H. Geubelle. 2017. Shape optimization using a NURBS-based interface-enriched generalized FEM. *Internat. J. Numer. Methods Engrg.* 111, 10 (jan 2017), 927–954.

B. Nayroles, G. Touzot, and P. Villon. 1992. Generalizing the finite element method: Diffuse approximation and diffuse elements. *Computational Mechanics* 10, 5 (1992), 307–318.

J. Nitsche. 1971. Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* 36, 1 (01 Jul 1971), 9–15.

Romain Prévost, Emily Whiting, Sylvain Lefebvre, and Olga Sorkine-Hornung. 2013. Make it stand. *ACM Trans. Graph.* 32, 4 (2013).

Masoud Safdari, Ahmad R. Najafi, Nancy R. Sottos, and Philippe H. Geubelle. 2015. A NURBS-based interface-enriched generalized finite element method for problems with complex discontinuous gradient fields. *Internat. J. Numer. Methods Engrg.* 101, 12 (jan 2015), 950–964.

Masoud Safdari, Ahmad R Najafi, Nancy R Sottos, and Philippe H Geubelle. 2016. A NURBS-based generalized finite element scheme for 3D simulation of heterogeneous materials. *J. Comput. Phys.* 318 (2016), 373–390.

Teseo Schneider, Yixin Hu, Jérémie Dumas, Xifeng Gao, Daniele Panozzo, and Denis Zorin. 2018. Decoupling Simulation Accuracy from Mesh Quality. *ACM Trans. Graph.* 37, 6 (Dec. 2018).

Adriana Schulz, Jie Xu, Bo Zhu, Changxi Zheng, Eitan Grinspun, and Wojciech Matusik. 2017. Interactive Design Space Exploration and Optimization for CAD Models. *ACM Trans. Graph.* 36, 4, Article 157 (July 2017), 14 pages.

Christian Schumacher, Jonas Zehnder, and Moritz Bächer. 2018. Set-in-stone: Worst-case Optimization of Structures Weak in Tension. *ACM Trans. Graph.* 37, 6, Article 252 (Dec. 2018), 13 pages.

Eftychios Sifakis and Jernej Barbič. 2012. FEM Simulation of 3D Deformable Solids: A Practitioner's Guide to Theory, Discretization and Model Reduction. In *ACM SIGGRAPH 2012 Courses (SIGGRAPH '12).* ACM, New York, NY, USA, Article 20, 50 pages.

F. S. Sin, D. Schroeder, and J. Barbič. 2013. Vega: Non-Linear FEM Deformable Object Simulator. *Computer Graphics Forum* 32, 1 (2013), 36–48.

Mélina Skouras, Bernhard Thomaszewski, Bernd Bickel, and Markus Gross. 2012. Computational Design of Rubber Balloons. *Comput. Graph. Forum* 31, 2pt4 (May 2012), 835–844.

Ondrej Stava, Juraj Vanek, Bedrich Benes, Nathan Carr, and Radomír Měch. 2012. Stress Relief: Improving Structural Strength of 3D Printable Objects. *ACM Trans. Graph.* 31, 4, Article 48 (July 2012), 11 pages.

Ivan E. Sutherland. 1963. Sketchpad: A Man-machine Graphical Communication System. In *Proceedings of the May 21-23, 1963, Spring Joint Computer Conference (AFIPS '63 (Spring)).* ACM, New York, NY, USA, 329–346.

Kiril Vidimče, Szu-Po Wang, Jonathan Ragan-Kelley, and Wojciech Matusik. 2013. OpenFab: A Programmable Pipeline for Multi-Material Fabrication. *ACM Transactions on Graphics* 32 (July 2013), 11. Issue 4.

L. Wang and E. Whiting. 2016. Buoyancy Optimization for Computational Fabrication. *Computer Graphics Forum* 35, 2 (may 2016), 49–58.

Jonas Zehnder, Espen Knoop, Moritz Bächer, and Bernhard Thomaszewski. 2017. Metasilicone. *ACM Trans. Graph.* 36, 6 (2017).

Yahan Zhou, Evangelos Kalogerakis, Rui Wang, and Ian R. Grosse. 2016. Direct shape optimization for strengthening 3D printable objects. *Computer Graphics Forum* 35, 7 (oct 2016), 333–342.