A framework for parametric design optimization using isogeometric analysis

Austin J. Herrema[a], Nelson M. Wiese[a,b], Carolyn N. Darling[a], Baskar Ganapathysubramanian[a], Adarsh Krishnamurthy[a], Ming-Chen Hsu[a,*]

[a]*Department of Mechanical Engineering, Iowa State University, 2025 Black Engineering, Ames, IA 50011, USA*
[b]*Departments of Physics & Mathematics, Central College, 812 University, Pella, Iowa 50219, USA*

Abstract

Isogeometric analysis (IGA) fundamentally seeks to bridge the gap between engineering design and high-fidelity computational analysis by using spline functions as finite element bases. However, additional computational design paradigms must be taken into consideration to ensure that designers can take full advantage of IGA, especially within the context of design optimization. In this work, we propose a novel approach that employs IGA methodologies while still rigorously abiding by the paradigms of advanced design parameterization, analysis model validity, and interactivity. The entire design lifecycle utilizes a consistent geometry description and is contained within a single platform. Because of this unified workflow, iterative design optimization can be naturally integrated. The proposed methodology is demonstrated through an IGA-based parametric design optimization framework implemented using the Grasshopper algorithmic modeling interface for Rhinoceros 3D. The framework is capable of performing IGA-based design optimization of realistic engineering structures that are practically constructed through the use of complex geometric operations. We demonstrate the framework's effectiveness on both an internally pressurized tube and a wind turbine blade, highlighting its applicability across a spectrum of design complexity. In addition to inherently featuring the advantageous characteristics of IGA, the seamless nature of the workflow instantiated in this framework diminishes the obstacles traditionally encountered when performing finite-element-analysis-based design optimization.

Keywords: Isogeometric analysis; Parametric design optimization; Rhino and Grasshopper; NURBS; Wind turbine blade

1. Introduction

One of the originally identified advantages of isogeometric analysis (IGA) [1] is that it enables tight integration of high-fidelity finite element analysis (FEA) into the engineering design work-

*Corresponding author
Email address: jmchsu@iastate.edu (Ming-Chen Hsu)

The final publication is available at Elsevier via http://dx.doi.org/10.1016/j.cma.2016.10.048

flow. In traditional design-and-analysis workflows, approximately 80% of the overall design lifecycle is devoted to finite element mesh generation and creation of analysis-suitable models; only 20% of the remaining lifecycle is spent on actually performing analysis [2]. In the context of iterative design optimization, such inefficiency is amplified. The core concept of IGA is the utilization of the geometric basis functions used to construct computer-aided design (CAD) models—usually non-uniform rational B-splines (NURBS)—directly as finite element basis functions, eliminating the need to generate an additional, geometrically approximate finite element mesh.

Previous works have sought to improve design-and-analysis frameworks by incorporating IGA. An isogeometric design-through-analysis concept was previously explored in Schillinger et al. [3] based on hierarchical refinement of NURBS and T-splines [2, 4] using the finite cell method [5, 6]. Breitenberger et al. [7] presented an Analysis in Computer-Aided Design (AiCAD) concept using NURBS-based boundary representation (B-rep) models for nonlinear isogeometric shell analysis, which included enhancements such as the ability to perform analysis on trimmed surfaces and the use of the penalty method for patch coupling. The AiCAD concept was implemented in CAD software packages such as Rhino [8] and Siemens NX [9]. Additionally, Hsu et al. [10] developed a user-interface-based parametric design platform for IGA directly within the Rhino CAD environment, utilizing Rhino's algorithmic modeling interface, Grasshopper [11], for parametric geometry generation.

The tight coupling between geometry and analysis also allows IGA to be naturally integrated with shape optimization. Wall et al. [12] and Fußeder et al. [13] presented frameworks for structural shape optimization of basic two-dimensional geometries using isogeometric structural analysis and gradient-driven optimization methods. Moysidis and Koumousis [14] performed shape optimization of plane stress structures in the context of a hysteric formulation for IGA. Julisson et al. [15] used IGA and Powell's derivative-free optimization algorithm to perform structural shape optimization of three-dimensional thin shell structures. Cho and Ha [16], Qian [17], and Kiendl et al. [18] used shape sensitivity analysis to recover optimal shapes through structural analysis, the latter doing so in three dimensions. In each of these cases, the locations of designated control points of interest (and control point weights, in some cases) were used as the design variables. Additionally, isogeometric shape optimization has been used to address problems of electromagnetic scattering [19], vibrating membranes [20], heat conduction [21], fluid mechanics [22], and the design of magnetic actuators [23]. A notable departure from the optimization of control point weights and locations is found in Kostas et al. [24] in which geometry parameterization is a primary focus and in which an IGA-based boundary element method provides the basis for optimizing the wave resistance of a T-spline ship hull.

While important work has been done in both uniting IGA with CAD software platforms and recognizing the natural ability of IGA to facilitate shape optimization, additional work must be

done to demonstrate that IGA-based optimization is relevant and beneficial in the context of modern CAD paradigms. For example, modern CAD systems, such as SolidWorks [25], Pro-Engineer [26], etc., use feature-based modeling[1] [29] to capture design intent [27]. In addition, they support parametric design modifications using a constraint-based solver [30–33] and solid modeling [34]. On the other hand, many isogeometric design-through-analysis frameworks are based on Rhino, a freeform surface modeling system, mainly because Rhino uses a NURBS-based geometry kernel that can be directly used for IGA. Rhino does not natively facilitate parametric constraint- or feature-based modeling. This makes editing engineering models in Rhino difficult, since model parameters cannot be simply changed to the desired value. The lack of feature-based modeling also implies that any change to a particular surface can lead to a geometry configuration that is inconsistent with the original design intent (e.g., introducing non-manifold geometry in a solid model, self-intersecting geometry, or new gaps). Hence, we may have to perform changes to multiple surfaces, even when we desire to change a single feature, in order to generate an analysis-suitable design. Finally, modern CAD platforms prioritize interactivity throughout the design process [35]. Thus, a parametric optimization workflow that uses design-specific, syntax-heavy, compilation-dependent code, while technically parametric, is not sufficiently interactive for extensive use in many engineering design contexts.

Another reality of realistic, large-scale engineering design contexts is that analyses from multiple disciplines must often be performed in order to quantify the effectiveness of a particular design, and the parametric inputs for these complex designs may be more abstract than fundamental geometric parameters such as control point locations (e.g., constraint-based dimensions or material parameters). Martins and Lambe [36] surveyed various methods encountered within the field of multidisciplinary design optimization, a field of research that studies the application of numerical optimization techniques to the design of engineering systems. Multidisciplinary design optimization is commonly used to address engineering design problems (see, e.g., [37–41]); some such problems use traditional FEA and are hence forced to script mesh generation procedures and to manage separate geometry descriptions.

The central goal of this work, therefore, is to develop a natural computational framework that is capable of IGA-based parametric design optimization in an interactive, multidisciplinary design context. In addition to facilitating high-level parameterization and interactivity, the framework should be designed around the notion of consistently creating families of analysis-suitable models. The methodology suggested in this work is demonstrated using the Grasshopper algorithmic mod-

[1] Please refer to the book by Shah and Mäntylä [27] and the review by Salomons et al. [28] for a detailed background on feature-based CAD. A feature tree is used to keep track of the different features that are used to model the geometry. A feature can give rise to new geometric entities, such as faces, which can then be used by a child feature in subsequent modeling steps.

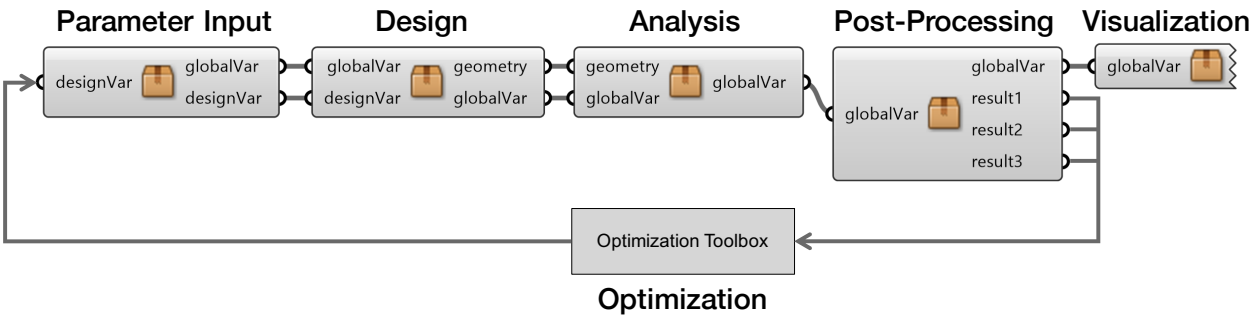

Figure 1: Overall structure of the isogeometric design optimization framework laid out in Grasshopper, an algorithmic modeling interface for Rhino. Optimization procedures can be performed either inside or outside of the Grasshopper environment using various optimization toolboxes, e.g. MATLAB [42], Dakota [43], Galapagos [11], etc.

eling interface to promote workflow consistency, efficiency, interactivity, and cost-effectiveness—in terms of both time and money—of the engineering design process. The Rhino-based Grasshopper can access Rhino's NURBS-based geometry kernel to construct realistic analysis-suitable models. The framework is also designed to enable truly seamless, heuristic design optimization based on IGA results. To the best of our knowledge, it is the first computational framework capable of performing IGA-based parametric design optimization of realistic engineering structures that are practically constructed through the use of complex, CAD-based geometric operations.

This paper is outlined as follows. In Section 2, we describe the structure of the isogeometric design optimization framework, highlighting the salient features of the parametric design, analysis, and optimization procedures. Specifically, we emphasize the role that each procedure plays in enabling the overall, novel approach to design optimization. In Section 3, we demonstrate the benefits and validate the capabilities of the framework by first optimizing a simple tube structure. We then optimize a wind turbine blade design in Section 4 to demonstrate the framework's ability to promote analysis-driven design of realistic, industrial-scale engineering solutions. In Section 5, we give our concluding remarks.

2. Isogeometric design optimization framework

In this work we develop a computational framework capable of performing IGA-based parametric design optimization of realistic engineering structures that are practically constructed through the use of complex, CAD-based geometric operations. The general structure of a simple design optimization procedure can be seen in Figure 1, which depicts the cycle of parametric model construction and analysis. Such cycles are commonplace in the engineering design world, demonstrated by the popularity of software platforms like ANSYS Workbench [44]. However, the use of specific strategies within the modeling and analysis stages of the design cycle, as proposed in this work and demonstrated below, allows seamless, IGA-based design optimization. Not only

is this seamless approach practical and efficient, but it also inherits the characteristics of IGA that are particularly advantageous in optimization settings (see Section 2.3.1 for more details).

The framework as implemented in this work and as shown in Figure 1 exists primarily within Grasshopper [11], an algorithmic modeling interface which makes use of and controls the CAD software called Rhino [8]. Rhino uses NURBS-based surface geometry descriptions and features a variety of numerically robust and efficient algorithms for creating and modifying NURBS geometry. Having access to this advanced geometric functionality is invaluable to engineering designers who rely heavily on complex, pre-defined algorithms. In the overarching context of the isogeometric design optimization framework, Grasshopper is used to create or integrate parametric design algorithms, analysis codes, post-processing operations, optimization toolboxes, and result visualization.

Grasshopper features many "components," which are visualized in the two-dimensional Grasshopper workspace as small rectangles, each with unique geometric or programmatic functions. The user inserts the desired components and links the functions' inputs and outputs together via graphical "wires." If no Grasshopper component contains the exact functionality desired by the user, custom scripting components, available in a variety of programming languages and capable of accessing Rhino's core functionality, can be created. Groups of functions can be also packaged into "clusters" which then appear as a single component in the Grasshopper workspace. The clusters responsible for the design, analysis, post-processing, and other operations are shown within the Grasshopper interface in Figure 1.

Subsequent sections will detail the contents of the design, analysis, and optimization components shown in Figure 1. Visualization of analysis results may not be required within the design optimization loop. However, it may be beneficial to visualize the results during or after the optimization process. We therefore detail the visualization methodology, which is unique for IGA and is implemented within the "Visualization" cluster in Figure 1, in Appendix A.

Remark 1. An interactive parametric design and geometry modeling platform was proposed in Hsu et al. [10] to directly employ IGA within the Rhino CAD environment. Hsu et al. [10] used a traditional approach to the model generation and analysis workflow in that the platform was constructed with the intent of a user interacting with the model within the Rhino viewport and invoking design, analysis and post-processing procedures via a user interface. This methodology is not suitable for rigorous design optimization in part because it was formulated with the intent of strong user interaction.

2.1. Design

The contents of the "Design" cluster in Figure 1 are necessarily distinct for unique design optimization applications. Here we discuss elements of engineering model design that are important

to consider when constructing an IGA framework intended to optimize realistic engineering designs. A few characteristics, such as parametric design, maintenance of valid analysis-suitable geometries, and interactivity, are considered indispensable for efficient model development.

2.1.1. Parametric model construction

The ability to establish direct parametric control of geometry is a necessity for most engineering designers. Most common CAD software, such as SolidWorks [25], employs constraint-based systems [45–47] that allow designers to directly alter model-defining dimensions such as line length or arc radius and geometric constraints like straightness or tangency. Changing any of these values causes the position and size of the relevant geometric entities to be automatically recalculated such that all user-defined constraints and dimensions are satisfied. This simplifies model construction and makes it easier to build design intent into a model, especially when the model is based on engineering drawings which use relative dimensioning almost exclusively.

Rhino is often used to generate models for IGA because it is built upon a NURBS-based geometry kernel and has been used within the IGA community in the past. However, Rhino does not natively feature constraint-based design capabilities, forcing the designer to painstakingly calculate the absolute position or size of geometric entities. Subsequent model adjustment must be performed in a similar fashion, rather than by the adjustment of the relevant model constraints or relations. This hinders the extent to which a designer can change an engineering design based on analysis results, a fundamental goal of IGA.

Another problem with using a freeform modeling system like Rhino in an iterative IGA context is that, due to the lack of constraint-based modeling capabilities, editing individual surfaces may lead to a model that is inconsistent with the original design intent. This behavior is demonstrated in Figure 2, where an initial model, on the left, is comprised of surfaces 1 and 2, where surface 2 is generated based on the location of the lower edge of surface 1. Using a freeform modeling system, if the designer were to change the radius of surface 1, then surface 2 would not be inherently regenerated accordingly. The result would be the top configuration on the right side of Figure 2, where the edges of the surfaces are no longer coincident.

We therefore require a modeling platform capable of both parametric modeling and consistent generation of geometry that aligns with the original design intent. In our framework, we utilize the Grasshopper interface for Rhino to achieve this goal. Grasshopper allows the designer to graphically develop a procedural algorithm [48] to create a model using interrelated geometric functions. The algorithms in our framework are developed such that the desired inputs are the engineering parameters of interest, effectively establishing direct parametric control of the NURBS objects within Rhino. Thus, procedural model generation gives the designer parametric control over the model without explicitly developing a fully developed, constraint-based modeling system.

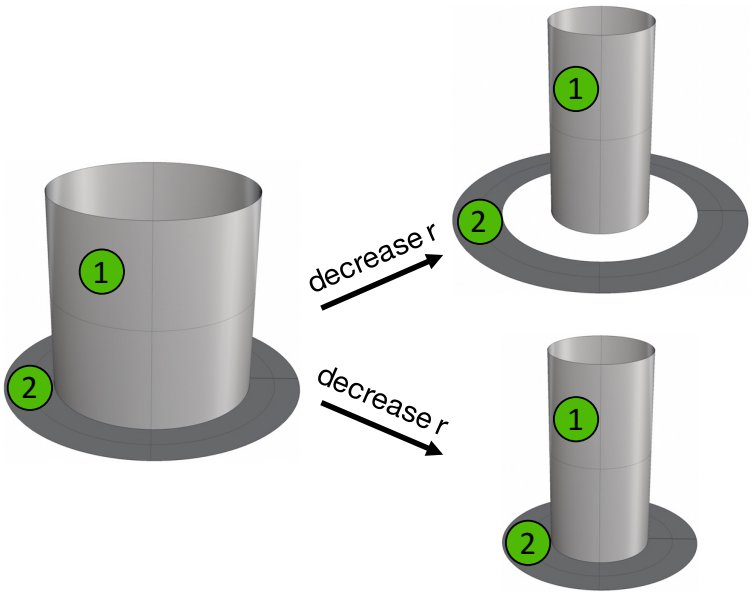

Figure 2: Illustration of the different designs achieved when design intent (geometric connection of surfaces 1 and 2) is maintained (bottom case) versus when design intent is not respected (top case) for a given model change.

The use of procedural generation also entails the capability to more reliably generate multiple geometries that maintain design intent. For the example in Figure 2, we can construct an algorithm wherein the radius of surface 1 is a parametric input and surface 2 is generated based on the lower edge of surface 1. Then, if the designer were to change the radius of surface 1, the entire algorithm would recompute, producing the configuration on the bottom-right side of Figure 2. Thus, the design remains valid for a wide range parametric input.

The notion of parametric procedural design can be abstracted using an expression $\Theta(\mathbf{x})$ in which $\mathbf{x}$ is a vector of parametric design variables and $\Theta(\mathbf{x})$ is an algorithm[2] that generates the design model based on given design variables. In this sense, the general expression $\Theta(\mathbf{x})$ might be thought of as a means of generating a "family of designs." Some common design variables, especially in the context of IGA, are the NURBS control point locations and weights; an example design vector $\mathbf{x}_e$ can thus be defined as

$$\mathbf{x}_e := \{\mathbf{P}_k, W_k\}, \quad k = 1, 2, \ldots, n\,, \tag{1}$$

where $\mathbf{P}$ and W denote the control point location and weight, respectively, boldface indicates a spatially dimensioned vector, and n is the total number of control points. We could then establish an algorithm, $\Theta_e(\mathbf{x}_e)$, for a particular design application. Of course, for the purposes of commu-

[2]In this context, we use the word "algorithm" to refer to a set of generative or manipulative geometric or otherwise programmatic functions which, when executed in sequence, procedurally generate a particular engineering design model.

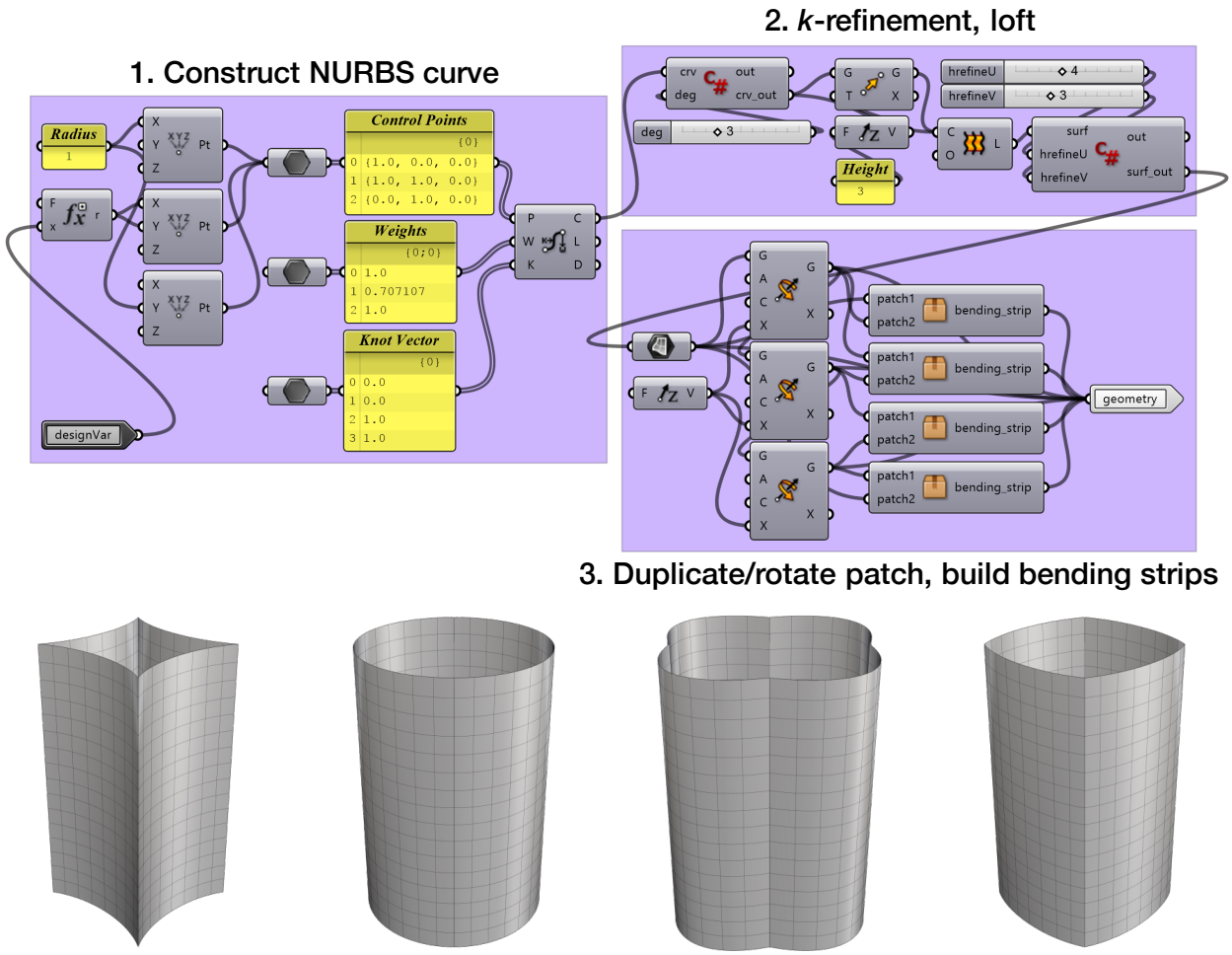

Figure 3: Family of tube designs (bottom) and associated Grasshopper generative algorithm $\Theta_t(\mathbf{x}_t)$ (top).

nication and efficiency, most realistic engineering designs require the establishment of high-level design parameters, such as component width or relative feature location.

Different sets of design parameters and generative algorithms correspond to different design scenarios. For example, the family of tube designs investigated in Section 3 and the Grasshopper algorithm $\Theta_t(\mathbf{x}_t)$ for generating these designs is shown in Figure 3. Additionally, the family of wind turbine blade designs investigated in Section 4 and the Grasshopper algorithm $\Theta_b(\mathbf{x}_b)$ for generating those designs is shown in Figure 4. More information on the design variables used in these cases can be found in the corresponding example sections.

2.1.2. Interactivity

Interactivity is important in the engineering design process not only because it improves the designer's aesthetic experience, but also because immediate visual feedback and intuitive interfaces improve the efficiency of the design process. This is yet another reason why we choose to demonstrate our computational framework in Grasshopper; rather than providing precise, robust

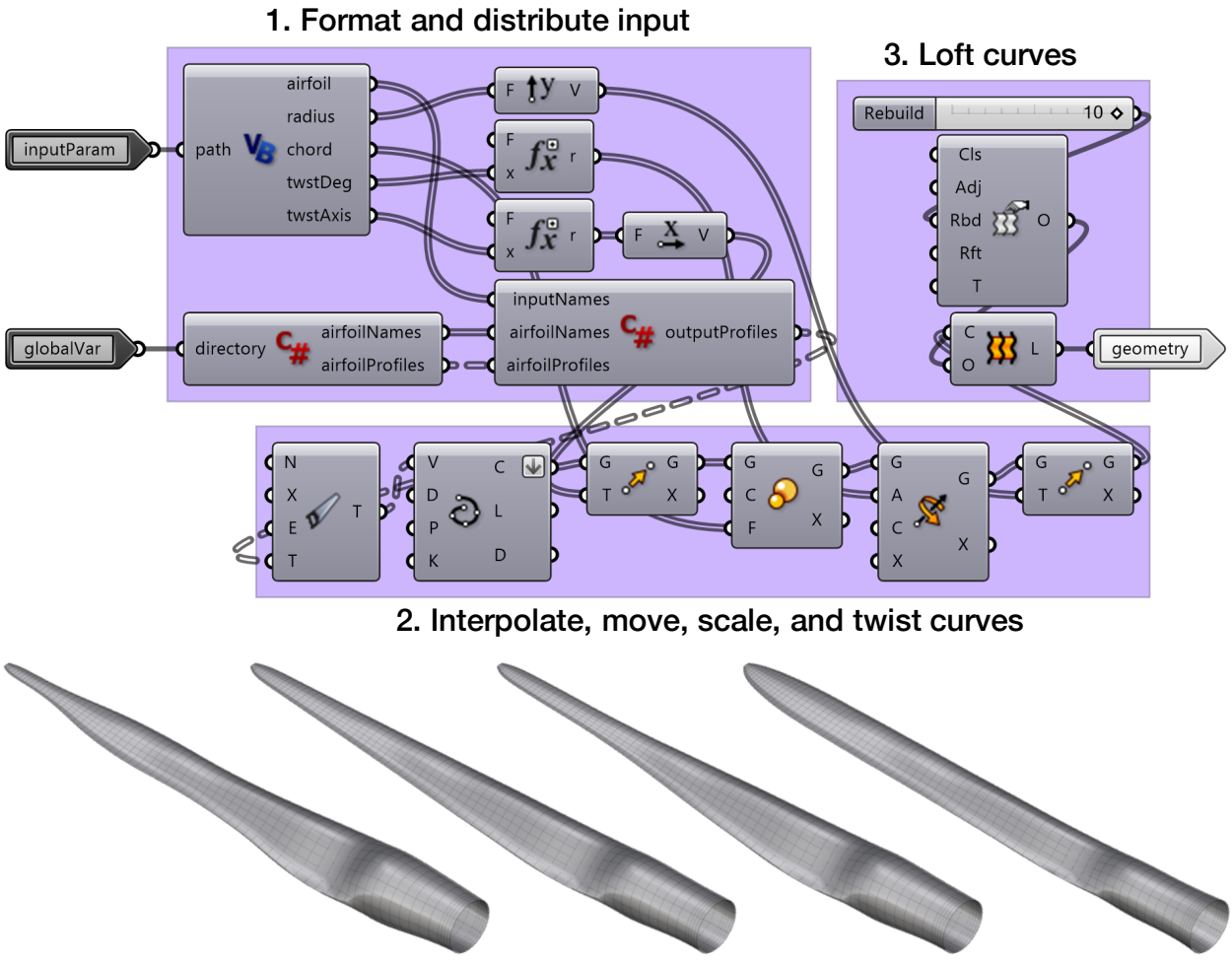

Figure 4: Family of blade designs (bottom) and associated Grasshopper generative algorithm $\Theta_b(\mathbf{x}_b)$ (top).

parametric design through syntax-heavy code, as is done in other works, the visually programmed generative algorithms are simple to edit, provide immediate visual feedback in three-dimensional space, and do not require compilation.

This point may seem trivial, but it is important to consider if we intend to abide by the original spirit of IGA. In essence, IGA and the notion of interactivity in the design context serve the same purpose: to improve the quality of design feedback and to deliver such feedback efficiently and elegantly. Thus, focusing on IGA without considering the interactive design context may not result in a net improvement in overall design-and-analysis workflow.

2.2. Analysis

Having established the utilization of a platform that is both NURBS-based and facilitates efficient, parametric model design, we incorporate IGA into the overall workflow as indicated by the "Analysis" cluster in Figure 1. Assuming a model's NURBS information, such as control point locations, degree, and knot vectors, is immediately available at the analysis stage, as is the case in

Grasshopper, and assuming that the geometry is analysis-suitable, IGA can be readily performed. The overall procedure in Grasshopper recognizes new parametric input, constructs the model according to a parametric algorithm, outputs the relevant NURBS information, and automatically calls an analysis code through a customizable C# scripting component. Compared to the overall analysis time, the computational cost of these input and output procedures is insignificant, as shown in Sections 3 and 4.

The applications currently of interest to the authors are relatively thin shell structures. For this reason, the rotation-free Kirchhoff–Love thin shell variational formulation is utilized for both the pressurized tube and the wind turbine blade applications in Sections 3 and 4. The isogeometric Kirchhoff–Love thin shell formulation was first proposed by Kiendl et al. [49] and further refined in Kiendl et al. [50] to handle regions where the mapping reduces to the C^0 level using the bending strip approach. The formulation was reformulated for composite shells in Bazilevs et al. [51] and was shown to accurately capture the dynamic kinematic behavior of wind turbine blades in Korobenko et al. [52] and Bazilevs et al. [53]. The formulation may be stated as: find the displacement of the shell midsurface $\mathbf{y} \in \mathcal{S}_y$ such that for all test functions $\mathbf{w} \in \mathcal{V}_y$,

$$\int_{\Gamma_0^s} \mathbf{w} \cdot h_{\text{th}} \overline{\rho}_0 \left(\frac{\mathrm{d}^2 \mathbf{y}}{\mathrm{d}t^2} - \mathbf{f} \right) \mathrm{d}\Gamma + \int_{\Gamma_0^s} \delta\overline{\boldsymbol{\varepsilon}} \cdot \left(\mathbf{K}_{\text{exte}} \overline{\boldsymbol{\varepsilon}} + \mathbf{K}_{\text{coup}} \overline{\boldsymbol{\kappa}} \right) \mathrm{d}\Gamma + \int_{\Gamma_0^s} \delta\overline{\boldsymbol{\kappa}} \cdot \left(\mathbf{K}_{\text{coup}} \overline{\boldsymbol{\varepsilon}} + \mathbf{K}_{\text{bend}} \overline{\boldsymbol{\kappa}} \right) \mathrm{d}\Gamma + \int_{\Gamma_0^b} \delta\overline{\boldsymbol{\kappa}} \cdot \mathbf{K}_{\text{best}} \overline{\boldsymbol{\kappa}} \, \mathrm{d}\Gamma - \int_{(\Gamma_0^s)_h} \mathbf{w} \cdot \mathbf{h} \, \mathrm{d}\Gamma = 0 \,, \tag{2}$$

where $\mathcal{S}_y$ and $\mathcal{V}_y$ denote the trial and test function spaces, respectively, for the structural mechanics problem, Γ_0^s and Γ_0^b denote the shell midsurface and bending strip domain in the reference configuration, respectively, h_{th} is the shell thickness, $\overline{\rho}_0$ is the through-thickness-averaged shell density, $\overline{\boldsymbol{\varepsilon}}$ and $\overline{\boldsymbol{\kappa}}$ are the membrane strain and curvature change of the midsurface, respectively, written in the local Cartesian system, $\delta\overline{\boldsymbol{\varepsilon}}$ and $\delta\overline{\boldsymbol{\kappa}}$ are their variations, $\mathbf{h}$ is the prescribed traction on $(\Gamma_0^s)_h$, $\mathbf{f}$ denotes body forces, $\mathbf{K}_{\text{exte}}$, $\mathbf{K}_{\text{coup}}$, and $\mathbf{K}_{\text{bend}}$ are the extensional, coupling, and bending stiffnesses, respectively, calculated using laminated plate theory, and $\mathbf{K}_{\text{best}}$ is the bending stiffness of the bending strips. (For more details, please see Bazilevs et al. [54].) We denote this weak form of the system of partial differential equations that describes the physics as $\mathcal{B}(\mathbf{y}) = 0$.

We emphasize that it would be entirely possible to replace the Kirchhoff–Love shell variational formulation used here with many other isogeometric methods, such as other shell formulations [55, 56], boundary element methods [57, 58], and finite cell [3, 5] or immersogeometric techniques [59, 60]. Much of the work in achieving such implementations would consist merely of ensuring that the IGA solver can recognize new geometries and communicate analysis results to the Grasshopper environment. It is also feasible to perform isogeometric analysis on solid volumetric geometries as was done with a gas turbine modeled using trivariate NURBS in Hsu et al. [10]. Because

Grasshopper and Rhino do not support trivariate splines natively, this is achieved through unique surface construction techniques and pre-processing that builds three-dimensional designs based on a network of two-dimensional surfaces. This represents a rich and fruitful avenue of future research and development.

2.3. Optimization

The "Optimization" cluster in Figure 1 indicates the use of an optimization toolbox to drive the iterative design-and-analysis process. In the introduction of this paper we discussed the notion that, although the fundamental integration of CAD and computer-aided engineering (CAE) paradigms through the use of IGA theoretically enables a more iterative approach to engineering design and analysis, practical limitations have hindered the establishment of IGA-based design-and-analysis workflows. These workflow limitations have led to limitations on how readily parametric design parameters can be optimized. One of the key goals of creating a parameterized design model within a design-through-analysis framework is to allow the designer to understand the influence of relevant design parameters on values of interest obtained through computational analysis. This can be done, and is often still done, manually, especially in the context of high-fidelity structural analysis; the designer performs analysis, views the result, adjusts the design, and repeats as necessary until the desired result is achieved. However, if the design-and-analysis workflow is made completely seamless using IGA and parametric design techniques, as is the case with the presented framework, we can further leverage computational power using automated optimization methods.

In this work, we integrate MATLAB into the design-and-analysis framework, allowing us to make use of the many optimization techniques incorporated into MATLAB's optimization toolbox [42]. Externally routing the design pipeline through MATLAB is acceptable in our implementation because relatively little information (in our cases, only design variables and cost function values) must be transferred. However, depending on the volume of transferred information and other performance requirements, alternative optimization techniques and packages, including techniques native to Grasshopper, could also be used. In our implementation, MATLAB provides input parameters, allows the design-through-analysis framework to build a model and perform analysis, and then retrieves relevant output values from Grasshopper to inform future iterations. This process is entirely automated and enables the optimization algorithm to search within a parameterized family of designs, $\Theta(\mathbf{x})$, freely. Importantly, both local (gradient-based and gradient-free) and global (meta-heuristic and multi-start) optimization methods can be seamlessly applied within this paradigm.

2.3.1. Advantages of IGA in optimization setting

As discussed previously, one traditional barrier to analysis automation in the finite element context is the difficulty associated with generating finite element meshes for complex geometries.

Ensuring that good quality meshes can be generated automatically from CAD models remains a challenging problem, often requiring manual intervention and thus reducing the overall efficiency of the optimization framework. A key benefit of a design optimization framework that makes use of IGA is that such mesh generation can be avoided, assuming the generative algorithm is designed carefully such that analysis suitability is ensured. The geometry can then be directly referenced for analysis, reducing the number of required pre-analysis tasks and easing setup of the overall optimization problem.

Isogeometric analysis may also reduce the computational time required for accurate analysis of a given design. This benefit is especially important in the context of design optimization, where the reduction of a single function evaluation by a minute can translate to hours of saved optimization time. Compared to traditional finite element methods, isogeometric analysis is capable of more quickly producing results of equivalent accuracy. Benson et al. [61] demonstrated that structural analysis of a roof using 450 quadratic NURBS elements could produce results in 2.90 CPU seconds that are approximately the same as those produced by an analysis using 4,512 linear Belytschko-Tsay elements, requiring 10.5 CPU seconds. It is therefore apparent that IGA is a uniquely apt tool in the context of design optimization, where limiting analysis time—without unnecessarily sacrificing analysis accuracy—is critical.

3. Tube profile optimization

In order to demonstrate the effectiveness of the IGA-based parametric design optimization framework, we first optimize a design with a known solution: the cross-sectional geometry of an internally pressurized tube.

3.1. Definition of cost function

We seek to solve the general optimization problem that is encoded in a cost functional $\mathcal{J}_t(\mathbf{y}; \mathbf{x}_t)$. The cost functional depends explicitly on the displacement field variables, $\mathbf{y}$, which are evaluated via solving the PDE, $\mathcal{B}(\mathbf{y}) = 0$. Additionally, the cost functional depends implicitly on the design variables $\mathbf{x}_t$ (usually via the field variables $\mathbf{y}(\mathbf{x}_t)$). The resulting PDE-constrained optimization problem is posed as follows:

$$\begin{aligned} &\text{minimize} && \mathcal{J}_t(\mathbf{y}; \mathbf{x}_t) \\ &\text{subject to} && \mathcal{B}(\mathbf{y}; \mathbf{x}_t) = 0 \,, \\ & && \mathbf{x}_t \in \mathbf{\Omega}_t \,. \end{aligned} \tag{3}$$

$\mathcal{J}_t(\mathbf{y}; \mathbf{x}_t)$, defined below, is calculated for each design-and-analysis iteration; $\mathbf{x}_t$ is the vector of design variables, defined in the succeeding section; and $\mathbf{\Omega}_t$ is the vector of allowable ranges for each design variable. Recall that, for the family of tube designs, we employ the generative algorithm

$\Theta_t(\mathbf{x}_t)$, shown in Figure 3, which acts as a preprocessor for the analysis of each design by producing the geometry definition that allows $\mathcal{J}_t(\mathbf{y}; \mathbf{x}_t)$ to be calculated.

For many structural analyses, as is the case here, it is reasonable to minimize the maximum strain in a design because strain is directly related to many popular failure criteria. In the isogeometric Kirchhoff–Love thin shell formulation [49, 62], the Green–Lagrange strain, $\overline{\mathbf{E}}$, is separated into a constant part, due to membrane action, and a linearly varying part, due to bending, as follows:

$$\overline{\mathbf{E}} = \bar{\boldsymbol{\varepsilon}} + \xi_3 \bar{\boldsymbol{\kappa}} \,, \tag{4}$$

where $\bar{\boldsymbol{\varepsilon}}$ denotes the membrane strain of the midsurface, $\bar{\boldsymbol{\kappa}}$ denotes the change in curvature of the midsurface due to bending, and ξ_3 is the through-thickness coordinate.

For this example the expected optimal cross-sectional shape is a circle because it is capable of supporting the entirety of the internal pressure load with only in-plane (membrane) stretching and zero bending action. We can therefore minimize

$$\mathcal{J}_t(\mathbf{y}; \mathbf{x}_t) = \bar{\boldsymbol{\kappa}}_{\max}(\mathbf{y}; \mathbf{x}_t) \,, \tag{5}$$

where $\mathbf{y}$ is the displacement and $\bar{\boldsymbol{\kappa}}_{\max}(\mathbf{y}; \mathbf{x}_t)$ is the maximum component of the maximum curvature change present in the design generated by the design variables $\mathbf{x}_t$.

3.2. Definition of design parameters

Much isogeometric shape optimization literature focuses on the optimization of control point locations (and weights, in some cases). While this is reasonable for small-scale problems like this one, it is desirable to reduce the number of design variables to produce only designs within a particular design space of interest. For the internally pressurized tube case we wish to constrain the design space such that it contains only tubes with a uniform cross-section and that are symmetric about two perpendicular planes. Therefore, as Figure 5 illustrates, we create one quarter of the cross-section using a NURBS curve featuring three control points with weights of $\{1, \frac{\sqrt{2}}{2}, 1\}$. The two end control points are fixed at a radial distance of one unit from the origin, and the middle control point is allowed to move radially towards or away from the origin. Therefore, for this problem, the design variables are defined as

$$\mathbf{x}_t \coloneqq \{r\} \,, \tag{6}$$

where r is the radial distance from the origin to the second NURBS control point as illustrated by Figure 5. This parameterization allows both square and circular cross-sections to be generated by varying a single design variable. The planar curve is then extruded in the perpendicular plane to generate a surface; the surface is duplicated and the duplicate surfaces are rotated to create the

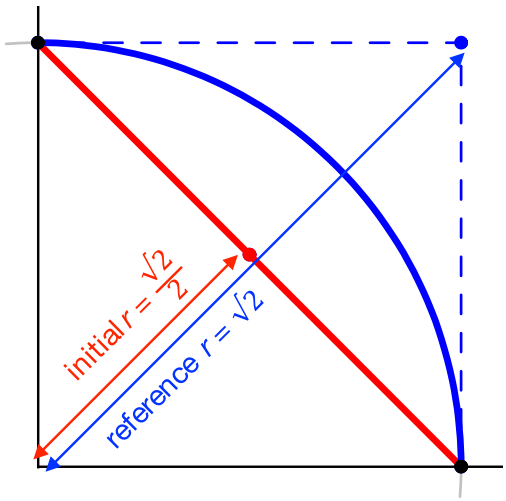

Figure 5: Design variable r definition for the internally pressurized tube problem.

remaining three quarters of the tube. The four quarters, each a single patch, are coupled using the bending strip method [50]. This generative geometric procedure, $\Theta_t(\mathbf{x}_t)$, is shown in Grasshopper in Figure 3.

3.3. Simulation setup and solution strategies

IGA mesh density is selected using k-refinement[3] (degree three in the u and v parametric directions) so as to balance the need for accuracy and the desire to limit analysis time, a critical factor for heuristic optimization techniques. A thickness of 2 cm is used with a Young's modulus of 0.4 GPa and Poisson's ratio of zero. The non-variable portions of the cross-sectional radius are fixed at 1 m and the height of the tube is 3 m. An internal pressure of 10 kPa is applied and the movement of a single control point is fixed for better numerical stability.

We directly apply Newton–Raphson iterations to converge the residual of this static problem. For each Newton–Raphson iteration, used to converge geometric nonlinearities, the linear system is solved using an iterative, diagonally preconditioned, conjugate gradient solver. Note that, in this case, there is only one design variable such that the cost function is one dimensional. A variety of optimization methods contained in the MATLAB Optimization Toolbox [42] can be used. Because we seek to demonstrate that the presented framework is applicable to a wide variety of engineering design problems, we use MATLAB's generalized pattern search (GPS) algorithm with positive 2N basis and mesh tolerance of 0.001. Pattern search methods do not suffer from some of the problems associated with gradient-based finite differencing, such as potential oversensitivity or

[3]The processes of knot insertion (h-refinement) and order elevation (p-refinement) do not commute. k-refinement, proposed in Hughes et al. [1], elevates the order of the original curve and then inserts a unique knot value. This process maintains the elevated-order continuity of the curve at the newly inserted knot.

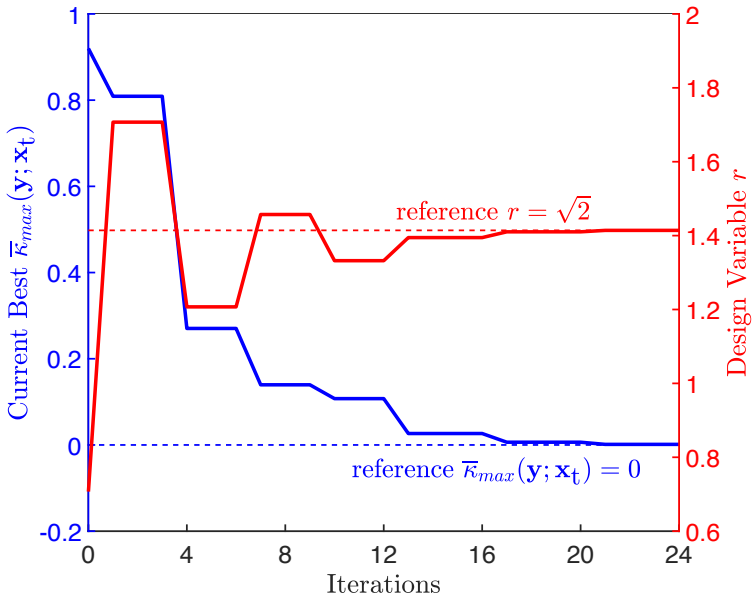

Figure 6: Design variable r and current best cost function value versus number of pattern search iterations.

insensitivity to design variable variation, and can therefore be more readily used for many design applications [42, 63], including problems of high dimensionality. Although this approach may not be the most efficient, it is reliable and also has rigorous local convergence properties [64].

3.4. Results and discussion

Optimization is performed using 16 GB RAM and a single core of a 2.2 GHz Intel Core i7 processor. A total of 38 designs are evaluated; each function evaluation takes about 25 seconds, yielding a total optimization time of about 16 minutes. Externally routing the optimization procedures through MATLAB requires the reading and writing of design variables and cost function variables, all of which requires less than one second for each function evaluation. Reading and writing analysis model data takes less than 100 ms per function evaluation. The entire process could theoretically be expedited by parallelizing structural analysis, optimization procedures, or both.

Figure 6 shows the design variable r plotted versus the cost function value, the maximum curvature change at any point in the design. It is clear that, throughout optimization, the design variable r converges towards the reference solution of $\sqrt{2}$, the radial control point position at which a perfectly circular cross-section is achieved. Additionally, the maximum curvature change, $\overline{\boldsymbol{\kappa}}_{\max}(\mathbf{y};\mathbf{x}_t)$, converges to zero.

These results are corroborated by Figure 7, which shows the strain contours on the undeformed and deformed geometries of the current best design at various points in the optimization process. Note that the maximum curvature change occurs in the initial, perfectly square cross-section, whereas after the last iteration, iteration 24, there is zero curvature change even after loading. The

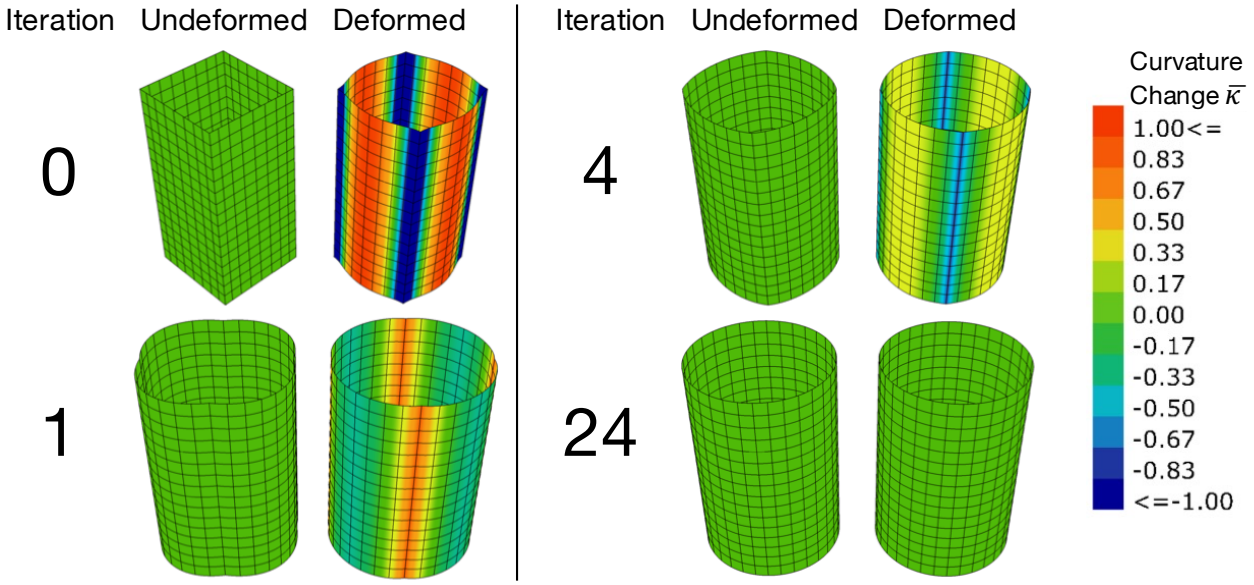

Figure 7: Undeformed and deformed shapes of current best tube design at selected optimization iterations. Color contour denotes value of maximum component of curvature change.

results are in good agreement with the expected values and demonstrate the framework's ability to optimize simple parametric designs using IGA and heuristic optimization techniques.

4. Wind turbine blade example

To demonstrate the applicability of the isogeometric design optimization framework to actual engineering problems, we use the proposed method to improve the design of the baseline NREL 5 MW wind turbine blade [65]. More specifically, we use the baseline design to establish a performance benchmark. We then use an optimization algorithm to vary a subset of design parameters to obtain a design with improved performance.

4.1. Definition of cost function

The optimization problem is posed as follows:

$$\begin{aligned} &\text{minimize} && \mathcal{J}_{\mathrm{b}}(\mathbf{y};\mathbf{x}_{\mathrm{b}}) \\ &\text{subject to} && \mathcal{B}(\mathbf{y};\mathbf{x}_{\mathrm{b}}) = 0\,, \\ & && \mathbf{x}_{\mathrm{b}} \in \boldsymbol{\Omega}_{\mathrm{b}}\,, \\ & && C_i(\mathbf{y};\mathbf{x}_{\mathrm{b}}) \leq 0\,, \qquad i = 1,\ldots,n_c\,, \end{aligned} \tag{7}$$

where $\mathcal{J}_{\mathrm{b}}(\mathbf{y};\mathbf{x}_{\mathrm{b}})$ is the cost function, defined below, calculated for each design-and-analysis iteration, $\mathbf{x}_{\mathrm{b}}$ is the vector of design variables, $\boldsymbol{\Omega}_{\mathrm{b}}$ is the vector of allowable ranges for the design variables, and there are n_c inequality constraints, C, that the optimized design must satisfy.

Formulating a meaningful cost function for complex engineering designs is challenging, but it is critical for achieving quality optimization results. In the case of wind turbine blades, a variety of values that derive from the creation and analysis of a computational model may be of interest to the designer. An effective cost function unites these values in a logical and meaningful way, essentially ranking the many design alternatives according to designer-defined objectives.

For this example, we quantify the effect that variation of the NREL 5 MW wind turbine blade design has on the machine's overall payback period, S, i.e. the amount of time that it takes for a wind turbine's total revenue production to match initial capital investment. Readers that are interested in the details of how the payback period, S, is formulated into the following cost function are referred to Appendix B. We define the cost function as

$$\mathcal{J}_{\mathrm{b}}(\mathbf{y};\mathbf{x}_{\mathrm{b}}) = \frac{1 + 0.1132\left(\frac{M(\mathbf{x}_{\mathrm{b}})-M_0}{M_0}\right)}{1 + \frac{P(\mathbf{x}_{\mathrm{b}})-P_0}{P_0}}, \tag{8}$$

where $M(\mathbf{x}_{\mathrm{b}})$ indicates the mass of a blade design variant, M_0 indicates the baseline NREL 5 MW blade design's mass, $P(\mathbf{x}_{\mathrm{b}})$ indicates the power production of a blade design variant, and P_0 indicates the baseline NREL 5 MW blade design's power production. The value of $\mathcal{J}_{\mathrm{b}}(\mathbf{y};\mathbf{x}_{\mathrm{b}})$ indicates a design alternative's payback period in terms of a proportion of the original payback period. The baseline 5 MW blade design has a $\mathcal{J}_{\mathrm{b}}(\mathbf{y};\mathbf{x}_{\mathrm{b}})$ of 1, or 100% the reference payback period; better performing designs have $\mathcal{J}_{\mathrm{b}}(\mathbf{y};\mathbf{x}_{\mathrm{b}}) < 1$; and poorer performing designs have $\mathcal{J}_{\mathrm{b}}(\mathbf{y};\mathbf{x}_{\mathrm{b}}) > 1$. Better performing designs will recover initial investment costs more quickly and should be more profitable overall. The right-hand side of (8) does not incorporate the displacements, $\mathbf{y}$, of the blade explicitly because the displacements are only used to calculate the constraints for this particular example.

4.2. Definition of design parameters and constraints

While simple geometries are often described using control point locations and weights, such fundamental geometry descriptions are intractable as primary descriptors of more complex models. More highly abstracted parametric relations are established for this reason. Wind turbine blades are generally constructed according to a set of design parameters that are defined at discrete locations along the blade. The geometric parameters are usually a section's radial location, chord length, airfoil shape, and twist degree. The Grasshopper algorithm for generating wind turbine blades is shown in Figure 4. For this simple optimization problem we focus on a single parameter, the chord length, which has definite implications for both blade mass and power production.

As can be seen in Jonkman et al. [65], the original 5 MW blade has nineteen locations, or "stations", at which the design parameters are defined, corresponding to nineteen chord lengths along the blade span. Rather than use these nineteen chord lengths as the design parameters for our opti-

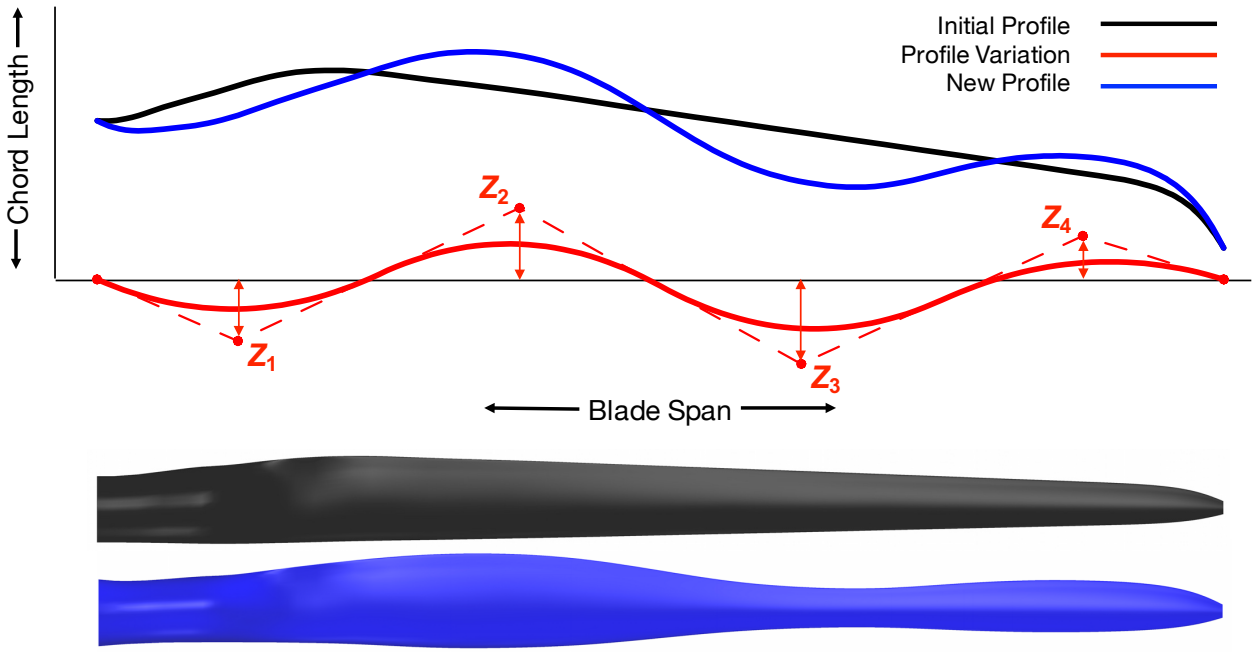

Figure 8: Demonstration of strategy for variation of chord profile using reduced number of parameters. Internal control points of a quadratic B-spline are moved and the variation profile is added to the original profile. Original blade geometry is shown in black (second from bottom) and new blade geometry is shown in blue (bottom).

mization problem, we use an alternative parameterization strategy to reduce the dimensionality of the design space. The strategy consists of creating a variation profile constructed from a quadratic B-spline of six control points evenly spaced along the blade span as shown in Figure 8. Varying the four internal control points in the direction of chord profile size allows semi-local control over this chord profile variation. The value of the variation profile at each of the nineteen cross-sectional locations along the blade span is added to the original profile to generate a new profile. The design variables for this example are therefore defined as

$$\mathbf{x}_\mathrm{b} \coloneqq \{Z_i\}, \quad i = 1, \ldots, 4\,, \tag{9}$$

where Z_i is the vertical movement in Figure 8 of each of the four internal control points. The generative algorithm $\Theta_\mathrm{b}(\mathbf{x}_\mathrm{b})$, excluding the algorithm for chord profile variation, is shown in Figure 4. The different blades in Figure 4 were generated using this variation approach.

An additional consideration for most realistic optimization problems is the optimization constraints. While the cost function $\mathcal{J}_\mathrm{b}(\mathbf{y}; \mathbf{x}_\mathrm{b})$ provides an explicit relationship between blade mass and power production, it does not take into account other potential constraints such as stress and strain or kinematics. We consider two such constraints in our studies: the maximum tip deflection of the blade, which is associated with tower clearance, and maximum in-plane strain, which is associated with material failure. Two constraint cases are optimized and discussed. For the first

case, we use a single constraint:

$$C_1(\mathbf{y}; \mathbf{x}_b) = \delta_{tip}(\mathbf{y}; \mathbf{x}_b) - \delta_{tip_0} \leq 0 \,, \tag{10}$$

where $\delta_{tip}(\mathbf{y}; \mathbf{x}_b)$ denotes the out-of-plane tip deflection of a potential blade design and δ_{tip_0} denotes the out-of-plane tip deflection of the baseline design. For the second constraint case, in addition to (10), we add

$$C_2(\mathbf{y}; \mathbf{x}_b) = \epsilon_{max}(\mathbf{y}; \mathbf{x}_b) - \epsilon_{max_0} \leq 0 \,, \tag{11}$$

where $\epsilon_{max}(\mathbf{y}; \mathbf{x}_b)$ denotes the maximum in-plane Green–Lagrange strain of a potential blade design and ϵ_{max_0} denotes the maximum in-plane Green–Lagrange strain of the baseline design. For both constraint cases, if any of the constraints are violated for a given set of design variables, the design variables are no longer considered potentially optimal solutions.

4.3. Simulation setup and solution strategies

Structural analysis is set up for the NREL 5 MW wind turbine blade and design variants $\Theta_b(\mathbf{x}_b)$ as follows. Because the geometry of wind turbine blades is critical to the machine's power production capability, we use Grasshopper to incorporate an aerodynamic analysis module, NREL's wind turbine analysis tool called FAST [66], which allows us to approximately calculate $P(\mathbf{x}_b)$ for a given design and to extract aerodynamic loads. FAST uses an implementation of blade element momentum theory to quickly produce an aerodynamic torque prediction for a given wind turbine setup. A force vector is calculated for each discrete segment (blade element) of the blade defined in FAST. These forces are distributed into traction vectors that are uniformly applied to the portion of the blade model corresponding to each FAST blade element. FAST is based on purely parametric input so it is easily incorporated into our framework. Typical blade design procedures require consideration of many different loading scenarios; however, for the purposes of this work, we consider a single loading scenario. We base our FAST analyses on a standard 5 MW setup described in Jonkman et al. [65] with a no-shear wind speed of 11.3 m/s, the speed at which the turbine should be operating at rated power and out-of-plane tip deflection should be relatively high.

For all blade designs, we define a simplified composite layup using some of the materials found in Sandia National Laboratory's composite layup definition for the NREL 5 MW blade [67]. Basic material zones—base layup, root, and spar cap—of uniform thickness are defined, shown in Figure 9. The entire blade surface consists of uni-directional E-LT-5500 fiberlass with additional uni-directional carbon in the spar cap region and SNLTriax added to the root. Material properties can be found in Resor [67]. Zone thicknesses are chosen such that the maximum tip deflection of the baseline blade design is approximately equal to the deflection specified in Jonkman et al. [65] under the given wind conditions. 3.25 cm of fiberglass, 1 cm of SNLTriax, and 8 cm of uni-

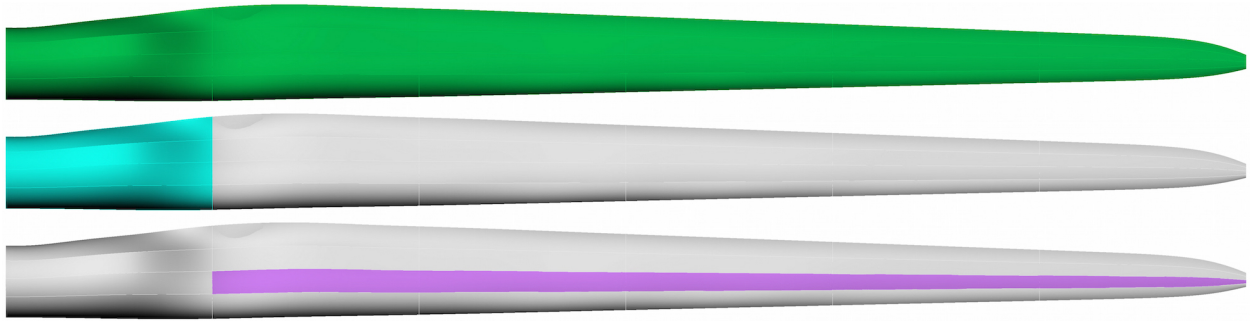

Figure 9: Simplified composite layup used for wind turbine blade optimization. Green color (top) indicates base E-LT-5500 fiberglass over entire blade, blue color (middle) indicates root buildup of SNLTriax, and purple color (bottom) indicates spar cap region made up of uni-directional carbon.

directional carbon are used. The total mass of the baseline blade design with this material setup, $M_0 = 40,912$ kg, is much higher than in the reference [65]; this is expected because the shear web structures are omitted for this simple example. A thicker shell definition is thus required to achieve realistic tip deflection under the given loading.

As in the previous example, IGA mesh density is selected using k-refinement (degree three in the u and v parametric directions) so as to balance the need for accuracy and the desire to reduce analysis time. Of course, if a higher degree of accuracy for each function evaluation is required, the mesh density can be increased, also increasing overall optimization time. For this dynamic problem, the algebraic problem is addressed by a direct application of Newton–Raphson iterations to converge the residual at each time step. As before, for each Newton–Raphson iteration, used to converge geometric nonlinearities, the linear system is solved using a diagonally preconditioned conjugate gradient method. The cost function is again minimized using MATLAB's generalized pattern search (GPS) algorithm with positive 2N basis and mesh tolerance of 0.01. Because we assume that the baseline 5 MW design should already have relatively good performance, we use the baseline 5 MW design, defined by $\mathbf{x}_b = 0$, as the initial point for pattern search optimization.

4.4. Results and discussion

As in the previous example, optimization is performed using a single core of a 2.2 GHz Intel Core i7 processor and 16 GB RAM. Each design evaluation takes approximately 9.5 minutes. Communication of information between MATLAB and Grasshopper takes less than one second per function evaluation, while the reading and writing of model data takes less than 100 ms per function evaluation. The total number of requisite function evaluations for the first and second constraint cases is 128 and 102, respectively. Thus, optimization using the first constraint case takes approximately 20 hours whereas optimization using the second constraint case takes approximately 16 hours. Again, the overall procedure could be expedited by parallelizing structural analysis, optimization procedures, or both. Tabular results of solution values of interest are shown in Table 1.

Table 1: Summary of results of interest for original and optimized designs. Only tip deflection is constrained for the first case, whereas both tip deflection and maximum strain are constrained for the second case. The overall payback period is reduced in both cases. Additional profit is defined over the entire lifetime of a large-scale offshore wind farm featuring an optimized blade design.

Design	Func. Evals	Tip Defl. (m)	Max. Strain	Mass (kg)	Power (kW)	$\mathcal{J}_b(\mathbf{y};\mathbf{x}_b)$	Add. Profit (millions \$)
Original	–	5.75	0.0083	40,912	5,265	100.00%	–
Case 1	128	5.75	0.0100	41,650	5,302	99.49%	6.37
Case 2	102	5.12	0.0083	43,265	5,311	99.78%	2.75

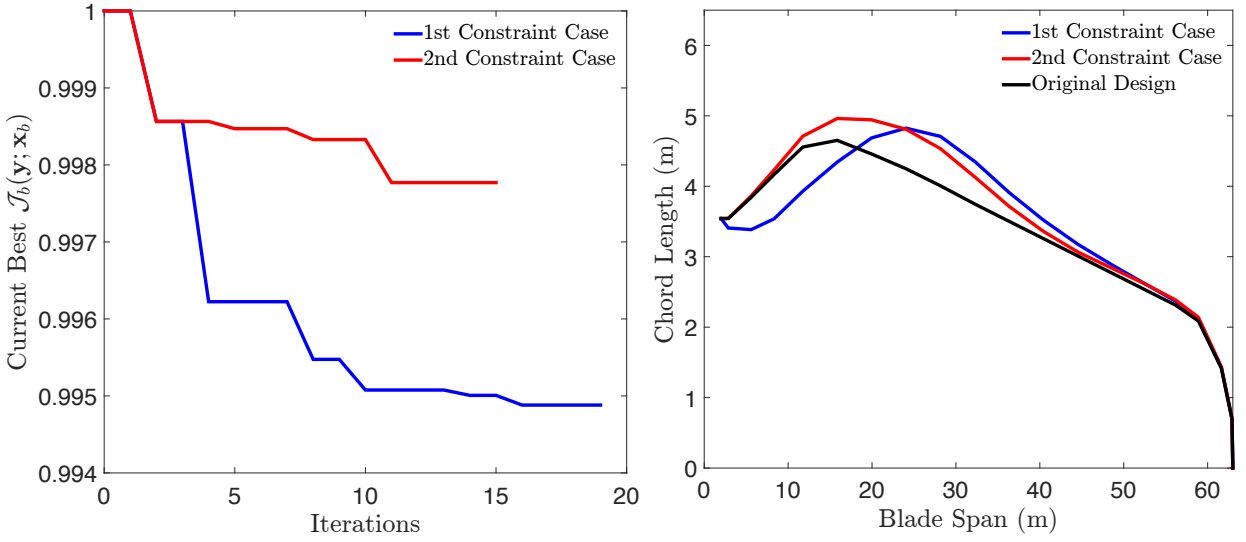

Figure 10: Iterative history of best cost function value for each constraint case (left) and chord profiles of original and optimized designs (right).

Graphs demonstrating both the optimization history of the cost function value and the optimized chord profiles for each constraint case are shown in Figure 10. The original and optimized blade geometries in the undeformed configuration are shown side by side in Figure 11. Comparison of strain distributions on original and optimized blade designs in their most deformed states are shown in Figure 12.

Table 1 shows that both optimizations yielded a design with a theoretical payback period slightly lower than the original payback period: a reduction of approximately 0.51% for the first constraint case and approximately 0.22% for the second constraint case, theoretically yielding an additional 6.38 and 2.75 million dollars (see Remark 2) of additional profit, respectively, over the lifetime of a large-scale wind farm. The difference between these two results is reasonable because the second case takes maximum strain, a potentially important factor depending on the design scenario, into account. Both optimized designs have larger overall profiles, increasing both mass and potential power output. Because power input increase is inversely related to the payback period in the cost function (8) and because it is weighted more heavily than mass, it is reasonable that an

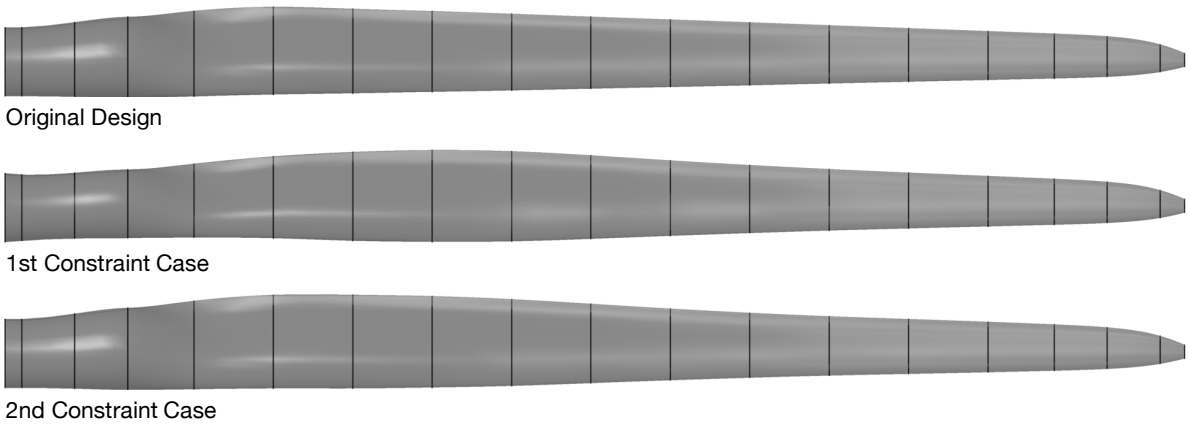

Figure 11: Comparison of original and optimized blade shapes viewed from the flapwise direction. Station sizes and locations are indicated by black lines.

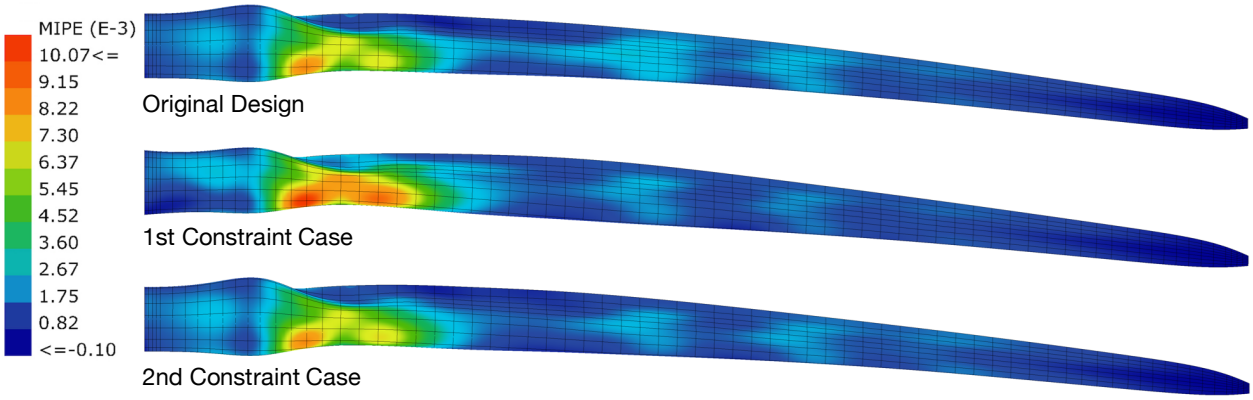

Figure 12: Comparison of strain distributions on original and optimized blade designs (shown in their most deformed states). The first principal in-plane strain on the outer surface of the shells is plotted. Blades are rotated 35 degrees from the flapwise direction used in Figure 11 to show region of strain concentration.

increase in both would be justifiable from a payback period perspective.

Remark 2. As Appendix B explains, using $\mathcal{J}_b(\mathbf{y}; \mathbf{x}_b)$ to calculate additional profit that is achievable over the life of a wind farm as a result of optimization, as is done in Table 1, requires additional assumptions to be made. The dollar values shown in this work are calculated using the Thornton Bank offshore wind farm, made up of 60 turbines with a 5 MW capacity and having a capital cost of 1.25 billion dollars, as a reference [68]. An offshore wind farm capacity factor of 42.4% is used [69] with an assumed price of electricity of 0.11 \$/kWh. As a basic performance measure, the simple payback period S ignores operations and maintenance costs.

The right plot in Figure 10 (right) provides greater insight into the salient design trends in these scenarios. The first optimized design has an increased chord length in the outer portion of the wind turbine blade's span, creating greater potential for aerodynamic torque production. Near the blade root, however, where the potential for aerodynamic torque production is the lowest, the blade's

chord size is reduced in an effort to reduce mass. Although there is a greater amount of force across the entirety of the blade and the root of the blade is smaller, the tip deflection is the same as the original because the thicker central portion of the blade provides additional stiffness. The increased maximum strain experienced by this aerodynamically aggressive design, however, may be undesirable.

The maximum in-plane strain is used as an additional constraint for the second optimization. The second optimized design, like the first optimized design, is generally wider to increase energy capture. However, rather than having a thinner root section, which ultimately increased stress concentration, the blade has a larger root. Thus, greater aerodynamic torque production is achieved without also increasing the maximum in-plane strain. Of course, mass is also higher in this case but, governed by the cost function (8), mass increase is offset by higher power production potential. Relative strain distributions and deformed blade shapes are shown in Figure 12.

The wind turbine blade example clearly demonstrates the benefits of using high-fidelity IGA and optimization in a realistic, complex design context. Without giving special attention or *a priori* "knowledge" to the system about particular design concepts that might be intuitive to human designers, the system is able to produce designs that align with human judgment—such as increasing chord size where energy capture potential is high or varying root size according to strain specifications—but in a more precise and less laborious way. Importantly, no effort is expended on finite element mesh generation throughout this entire design process.

5. Concluding Remarks

We presented a computational framework for parametric design optimization using isogeometric analysis. In Section 2 we summarized the principal features of the computational framework and emphasized the role of each feature in enabling a novel IGA-based parametric design optimization methodology. The framework is based on Grasshopper, an algorithmic modeling interface that abides by and uniquely integrates a number of important design philosophies and that also contains powerful geometry manipulation functions that enable the parametric generation of models suitable for IGA. In the context of this unified design framework, which features consistent geometry descriptions throughout design and analysis, analysis-driven optimization even of complex designs is natural and relatively simple. It is a unique framework in that it enables parametric design optimization of a variety of CAD-generated engineering structures using IGA.

In Section 3, we demonstrate the framework's ability to accurately optimize a simple pressurized tube design, a design parameterized with a single design variable. In Section 4, we consider a more realistic design scenario with more highly abstracted design parameters: the design of a wind turbine blade. Optimizing a different design does not require fundamental restructuring of the isogeometric design optimization framework. Instead, optimization merely requires the development

of the Grasshopper design algorithm $\Theta_b(\mathbf{x}_b)$ for wind turbine blades, a relatively simple task for experienced designers made simpler via the use of an interactive interface, and the selection and integration of an appropriate IGA methodology. We optimize the theoretical payback period of a 5 MW wind turbine according to a variation of the wind turbine blade design. We show that, under the given assumptions, the payback period could be reduced by approximately 0.22% in the most conservatively constrained optimization case. Even this small percentage improvement could yield an additional profit on the order of 2.75 million dollars over the life of a large-scale offshore wind farm. The optimization also reveals analysis-based trends which are useful to the designer. In addition, consistent with the fundamental goals of IGA, no effort is expended on traditional finite element mesh generation throughout the entire design process.

Overall, this framework demonstrates how the benefits of IGA can be leveraged in realistic engineering design contexts to generate optimized designs and design alternatives based on high-fidelity structural analysis, reducing designer labor. One of the fundamental goals of computational analysis and design is, simply stated, to achieve optimized designs before experimentation or production even begins. In actual practice, however, the state of the modern engineering workflow is a significant barrier to the realization of this goal. This work directly addresses not only the problem of design and analysis, but the design-and-analysis environment itself. Addressing the issues encountered in this context represents an important step towards enabling more effective use of IGA-based parametric design optimization.

Acknowledgements

A.J. Herrema was supported by the U.S. National Science Foundation (NSF) Grant No. DGE-1069283 which funds the activities of the Integrative Graduate Education and Research Traineeship (IGERT) in Wind Energy Science, Engineering, and Policy (WESEP) at Iowa State University. N.M. Wiese was supported by the NSF Grant No. EEC-1263243 which funds the activities of Research Experiences for Undergraduates (REU) in the area of Microscale Sensing, Actuation and Imaging (MoSAIc) at Iowa State University. M.-C. Hsu was partially supported by the ARO Grant No. W911NF-14-1-0296. B. Ganapathysubramanian was partially supported by the NSF Grant No. CMMI-1404938. A. Krishnamurthy was partially supported by the NSF Grant No. CMMI-1644441. This support is gratefully acknowledged.

Appendix A. Visualization of IGA results

Visualizing the simulation results of structural analyses throughout optimization can provide valuable feedback and can help the designer to understand the progression of the optimization procedure. This is especially relevant if a solution field, such as maximum in-plane strain, is used

as an optimization constraint or objective. After solving the IGA simulation, the control variables (or degrees of freedom) for the solution fields (e.g., displacement, velocity, temperature, etc.) are defined on the control points, which are typically not located on the physical geometry. These need to be coupled with basis functions to generate continuous solution fields that can be mapped to the physical geometry. In this work we use a simple Grasshopper-generated visualization mesh to map the solution fields. More sophisticated visualization techniques, such as direct volume rendering [70, 71], isosurface mesh extraction [72, 73], and direct rendering of isosurfaces [74–76] have been developed for visualizing volumetric IGA results.

An approximate, mesh-based methodology for visualizing IGA results within the Rhino viewport was proposed in Hsu et al. [10]. A visualization mesh is constructed and the coordinates of the mesh points are fed to a Grasshopper component that finds their closest points on the NURBS surface and returns the parametric coordinates of these closest points. Along with control variables, control points and basis function information, the solution values are evaluated using an in-house code and then transferred back to the visualization mesh points.

In this work, we propose an entirely Grasshopper-based implementation of this approach in which we construct a "solution surface" within Rhino's geometry kernel which is evaluated at the mesh points. This is possible because we are performing analysis only on thin-shell structures. Grasshopper natively features visualization meshes for the display of color contours on geometries. A relatively dense visualization mesh can easily be generated for virtually any geometry. A color can then be assigned to each mesh point, defined by parametric (u, v) coordinates. Thus, we wish to evaluate a particular solution parameter, such as maximum in-plane strain, at each mesh point location defined by parametric (u, v) coordinates. Also, we note that, in IGA, solution coefficients may be assembled to each control point; we denote these solution coefficients $\mathbf{Q}_{i,j}$, where i and j correspond to the index of the control point in the u and v directions, respectively.

A NURBS surface of degree p in the u direction and degree q in the v direction has the form

$$\mathbf{S}(u, v) = \sum_{i=1}^{n_c} \sum_{j=1}^{m_c} R_{i,j}^{p,q}(u, v)\mathbf{P}_{i,j} \,, \tag{A.1}$$

where the basis function $R_{i,j}^{p,q}(u, v)$ is defined over the (u, v) parametric space, n_c and m_c are the total number of control points in the u and v directions, respectively, and a net of control points is given by $\mathbf{P}_{i,j}$. Further details regarding the calculation of $R_{i,j}^{p,q}(u, v)$ can be found in Piegl and Tiller [77]. Rhino's C# programming library [78], which can be referenced by the C# scripting components in Grasshopper, contains a function for the construction of a NURBS surface $\mathbf{S}(u, v)$, given the constituents of the basis function $R_{i,j}^{p,q}(u, v)$ (such as knot vectors, surface degrees, and control weights) and the control points $\mathbf{P}_{i,j}$. Because we seek to evaluate our solution, rather than the geometry's physical location, at the given mesh points, we can utilize these same C# functions

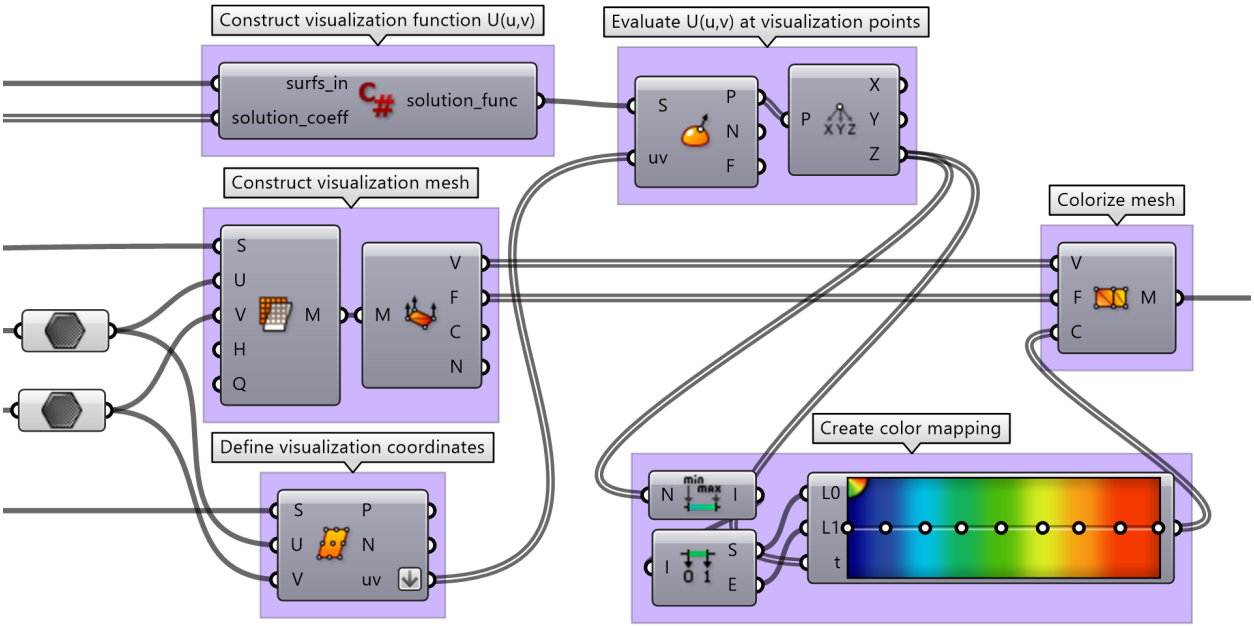

Figure A.13: Grasshopper implementation of visualization methodology.

to construct *not* a physical surface $\mathbf{S}(u, v)$ but rather a mapping of our solution variable:

$$\mathbf{U}(u, v) = \sum_{i=1}^{n_c} \sum_{j=1}^{m_c} R_{i,j}^{p,q}(u, v)\mathbf{Q}_{i,j} \,, \tag{A.2}$$

where $R_{i,j}^{p,q}(u, v)$ is exactly the same as in (A.1), but the solution coefficients $\mathbf{Q}_{i,j}$ are used in place of the control points $\mathbf{P}_{i,j}$. This process constructs a solution "surface" $\mathbf{U}(u, v)$, which may be evaluated for each mesh point. Having obtained a result value for each mesh point, the values are assigned a color according to a relative color scale and then visualized in the Rhino viewport via the visualization mesh.

The Grasshopper implementation of this process is shown in Figure A.13. The solution mapping, $\mathbf{U}(u, v)$, is constructed within the C# scripting component in the top left, whereas it is evaluated at the mesh point coordinates in the components in the upper right. The solution mesh is constructed in the components in the bottom left of the figure and is then colorized according to the evaluated solution values using the bottom right components. This colorized solution mesh is then automatically displayed in the Rhino viewport. The density of the visualization mesh can also be varied within the Grasshopper definition since $\mathbf{U}(u, v)$ itself is interactively evaluated within the definition.

Appendix B. Formulation of wind turbine blade cost function

The cost function used in the wind turbine blade example is formulated as follows. We first consider the following relation, a relatively common measure used in the wind energy industry:

$$S = \frac{K_{\mathrm{CC}}}{K_{\mathrm{AAR}}}, \tag{B.1}$$

where K_{CC} is the total capital cost of the machine, K_{AAR} is the average annual return, and S is the simple payback period for the machine [79]. The simple payback period, as implied by this definition, is the amount of time that it takes for a wind turbine's total revenue production to match initial capital investment. A reduction of payback period indicates that a turbine will be able to produce profit over a larger portion of its operating life.

Rather than computing the full numerical capital cost and average annual return for every blade design, we can instead approximately quantify the effect a particular variation of the baseline design would have on the simple payback period using some assumptions. We consider the following equation, defined for each blade design variant:

$$S(\mathbf{x}_{\mathrm{b}}) = \frac{K_{\mathrm{CC}_0} + V_{\mathrm{CC}}(\mathbf{x}_{\mathrm{b}})K_{\mathrm{CC}_0}}{K_{\mathrm{AAR}_0} + V_{\mathrm{AAR}}(\mathbf{x}_{\mathrm{b}})K_{\mathrm{AAR}_0}} = C_0\frac{1 + V_{\mathrm{CC}}(\mathbf{x}_{\mathrm{b}})}{1 + V_{\mathrm{AAR}}(\mathbf{x}_{\mathrm{b}})}, \tag{B.2}$$

where the subscript zero on K_{CC} and K_{AAR} indicates reference values that are obtained from analysis of a baseline blade design, $V_{\mathrm{CC}}(\mathbf{x}_{\mathrm{b}})$ indicates the fractional variation of the capital cost as a result of design variation, and $V_{\mathrm{AAR}}(\mathbf{x}_{\mathrm{b}})$ indicates the fractional variation of the average annual return as a result of design variation. The constant C_0 entails all components of the original capital cost, K_{CC_0}, and original average annual return, K_{AAR_0}, which are unaffected by the blade design variation $\mathbf{x}_{\mathrm{b}}$. The design-dependent values that most directly influence the simple payback period are the blade's mass and power output; mass is related to $V_{\mathrm{CC}}(\mathbf{x}_{\mathrm{b}})$ in the numerator of Eq. (B.2) and power is related to $V_{\mathrm{AAR}}(\mathbf{x}_{\mathrm{b}})$ in the denominator.

We first consider the numerator and the influence of mass variation. In [80] the International Renewable Energy Agency (IRENA) states that, for 5 MW applications, the blades make up 22.2% of the capital cost of the wind turbine. It further states the capital cost of the turbine itself comprises 51% of the total capital cost of offshore wind turbine installations; combining these claims, we surmise that 11.32% of the total capital cost is due to the blades. We recognize that other sources may cite varying percentages, but the value of 11.32% is sufficient for this example. If we assume that the mass of the blade is proportional to its cost, we can formulate the variation in the capital cost due to blade variation as follows:

$$V_{\mathrm{CC}}(\mathbf{x}_{\mathrm{b}}) = 0.1132\left(\frac{M(\mathbf{x}_{\mathrm{b}}) - M_0}{M_0}\right), \tag{B.3}$$

where $M(\mathbf{x}_\mathrm{b})$ indicates the mass of a blade design variant and M_0 indicates the baseline NREL 5 MW design's mass.

We now consider the denominator of Eq. (B.2), containing V_AAR. The average annual return, K_AAR, is equal to $E_a D_e$, where E_a is the annual energy production and D_e is the price obtained for electricity. We may further recognize that the annual energy production E_a is the product of the nameplate capacity of a machine, P, and the capacity factor, C_F. Thus, $K_\mathrm{AAR} = PC_F D_e$. Because we are considering blade design variants that might be used interchangeably in the same operating environment, we consider D_e and C_F to be constant and lump them into C_0, allowing us to formulate the variation in average annual return due to blade variation as

$$V_\mathrm{AAR}(\mathbf{x}_\mathrm{b}) = \frac{P(\mathbf{x}_\mathrm{b}) - P_0}{P_0} \,, \tag{B.4}$$

where $P(\mathbf{x}_\mathrm{b})$ indicates the power production of a blade design variant and P_0 indicates the baseline NREL 5 MW design's power production. Substituting Eqs. (B.3) and (B.4) into Eq. (B.2) we are left with

$$S(\mathbf{x}_\mathrm{b}) = C_0 \frac{1 + 0.1132\left(\frac{M(\mathbf{x}_\mathrm{b}) - M_0}{M_0}\right)}{1 + \frac{P(\mathbf{x}_\mathrm{b}) - P_0}{P_0}} \,. \tag{B.5}$$

Because we desire to minimize Eq. (B.5) and because C_0, a constant, is proportionally related to the rest of the equation, we can finally define our cost function as the non-constant portion of Eq. (B.5), or

$$\mathcal{J}_\mathrm{b}(\mathbf{y}; \mathbf{x}_\mathrm{b}) = \frac{1 + 0.1132\left(\frac{M(\mathbf{x}_\mathrm{b}) - M_0}{M_0}\right)}{1 + \frac{P(\mathbf{x}_\mathrm{b}) - P_0}{P_0}} \,. \tag{B.6}$$

Note that the right-hand side of Eq. (B.6) does not incorporate the blade displacements, $\mathbf{y}$, explicitly because the displacements are only used to calculate the constraints for this particular example.

The simple payback period, S, is not the only metric used to judge overall cost efficiency of wind turbines. Other more sophisticated metrics include the cost of energy (COE) and levelized cost of energy (LCOE) and could be used in a similar fashion. The simple payback period is used as a basic demonstration of a multidisciplinary objective.

In our efforts to limit the scope of this optimization problem we ignore certain factors that would not be superfluous in an actual blade design context. Such factors include the effect of blade design on power production across the entire possible range of wind conditions, the effect of blade mass on tower cost, and modal changes due to mass redistribution.

References

[1] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194:4135–4195, 2005.

[2] Y. Bazilevs, V. M. Calo, J. A. Cottrell, J. A. Evans, T. J. R. Hughes, S. Lipton, M. A. Scott, and T. W. Sederberg. Isogeometric analysis using T-splines. *Computer Methods in Applied Mechanics and Engineering*, 199:229–263, 2010.

[3] D. Schillinger, L. Dedè, M. A. Scott, J. A. Evans, M. J. Borden, E. Rank, and T. J. R. Hughes. An isogeometric design-through-analysis methodology based on adaptive hierarchical refinement of NURBS, immersed boundary methods, and T-spline CAD surfaces. *Computer Methods in Applied Mechanics and Engineering*, 249–252:116–150, 2012.

[4] X. Wei, Y. Zhang, L. Liu, and T. J. R. Hughes. Truncated T-splines: Fundamentals and methods. *Computer Methods in Applied Mechanics and Engineering*, 2016. http://dx.doi.org/10.1016/j.cma.2016.07.020.

[5] E. Rank, M. Ruess, S. Kollmannsberger, D. Schillinger, and A. Düster. Geometric modeling, isogeometric analysis and the finite cell method. *Computer Methods in Applied Mechanics and Engineering*, 249-252:104–115, 2012.

[6] D. Schillinger and M. Ruess. The Finite Cell Method: A review in the context of higher-order structural analysis of CAD and image-based geometric models. *Archives of Computational Methods in Engineering*, 22(3):391–455, 2015.

[7] M. Breitenberger, A. Apostolatos, B. Philipp, R. Wüchner, and K.-U. Bletzinger. Analysis in computer aided design: Nonlinear isogeometric B-Rep analysis of shell structures. *Computer Methods in Applied Mechanics and Engineering*, 284:401–457, 2015.

[8] Rhino. http://www.rhino3d.com/. Accessed 27 May 2016.

[9] Siemens NX. https://www.plm.automation.siemens.com/en_us/products/nx/. Accessed 27 May 2016.

[10] M.-C. Hsu, C. Wang, A. J. Herrema, D. Schillinger, A. Ghoshal, and Y. Bazilevs. An interactive geometry modeling and parametric design platform for isogeometric analysis. *Computers and Mathematics with Applications*, 70:1481–1500, 2015.

[11] Grasshopper. http://www.grasshopper3d.com/. Accessed 27 May 2016.

[12] W. A. Wall, M. A. Frenzel, and C. Cyron. Isogeometric structural shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 197:2976–2988, 2008.

[13] D. Fußeder, B. Simeon, and A.-V. Vuong. Fundamental aspects of shape optimization in the context of isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 268:313–331, 2015.

[14] A. N. Moysidis and V. K. Koumousis. A hysteric formulation for isogeometric analysis and shape optimization of plane stress structures. In *8th GRACM International Congress on Computational Mechanics*, Volos, Greece, 2015.

[15] S. Julisson, C. Fourcade, P. de Nazelle, and L. Dumas. A novative optimal shape design based on an isogeometric approach: Application to optimization of surface shapes with discontinuous curvature. In *11th World congress on structural and multidisciplinary optimization (WCSMO-11)*, Sydney, Australia, 2015.

[16] S. Cho and S.-H. Ha. Isogeometric shape design optimization: exact geometry and enhanced sensitivity. *Structural and Multidisciplinary Optimization*, 38:53–70, 2009.

[17] X. Qian. Full analytical sensitivities in NURBS based isogeometric shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 199:2059–2071, 2010.

[18] J. Kiendl, R. Schmidt, R. Wüchner, and K.-U. Bletzinger. Isogeometric shape optimization of shells using semi-analytical sensitivity analysis and sensitivity weighting. *Computer Methods in Applied Mechanics and Engineering*, 274:148–167, 2014.

[19] D. M. Nguyen, A. Evgrafov, and J. Gravesen. Isogeometric shape optimization for electromagnetic scattering problems. *Progress in Electromagnetics Research B*, 45:117–146, 2012.

[20] N. D. Manha, A. Evgrafov, A. R. Gersborg, and J. Gravesen. Isogeometric shape optimization of vibrating membranes. *Computer Methods in Applied Mechanics and Engineering*, 200: 1343–1353, 2011.

[21] M. Yoon, M.-J. Choi, and S. Cho. Isogeometric configuration design optimization of heat conduction problems using boundary integral equation. *International Journal of Heat and Mass Transfer*, 89:937–949, 2015.

[22] P. Nørtoft and J. Gravesen. Isogeometric shape optimization in fluid mechanics. *Structural and Multidisciplinary Optimization*, 48(5):909–925, 2013.

[23] S.-W. Lee, J. Lee, and S. Cho. Isogeometric shape optimization of ferromagnetic materials in magnetic acuators. *IEEE Transactions on Magnetics*, 52(2):1–8, 2016.

[24] K. V. Kostas, A. I. Ginnis, C. G. Politis, and P. D. Kaklis. Ship-hull shape optimization with a T-spline based BEM-isogeometric solver. *Computer Methods in Applied Mechanics and Engineering*, 284:611–622, 2015.

[25] SolidWorks. http://www.solidworks.com/. Accessed 27 May 2016.

[26] Pro/ENGINEER. http://www.ptc.com/cad/pro-engineer. Accessed 13 Oct 2016.

[27] J. J. Shah and M. Mäntylä. *Parametric and feature-based CAD/CAM: concepts, techniques, and applications*. John Wiley & Sons, 1995.

[28] O. W. Salomons, F. J. A. M. van Houten, and H. J. J. Kals. Review of research in feature-based design. *Journal of manufacturing systems*, 12(2):113–132, 1993.

[29] L. K. Kyprianou. *Shape classification in computer-aided design*. PhD thesis, University of Cambridge, 1980.

[30] A. Verroust, F. Schonek, and D. Roller. Rule-oriented method for parameterized computer-aided design. *Computer-Aided Design*, 24(10):531–540, 1992.

[31] I. E. Sutherland. Sketch pad a man-machine graphical communication system. In *Proceedings of the SHARE design automation workshop*, pages 6–329. ACM, 1964.

[32] H. Suzuki, H. Ando, and F. Kimura. Geometric constraints and reasoning for geometrical cad systems. *Computers & Graphics*, 14(2):211–224, 1990.

[33] A. Borning. ThingLab: an object-oriented system for building simulations using constraints. In *Proceedings of the 5th international joint conference on Artificial intelligence-Volume 1*, pages 497–498. Morgan Kaufmann Publishers Inc., 1977.

[34] A. AG Requicha and H. B. Voelcker. Constructive solid geometry. Technical report, University of Rochester, 1977.

[35] A. Krishnamurthy. *Parallel GPU Algorithms for Mechanical CAD*. PhD thesis, University of California, Berkeley, 2010. URL http://eprints.cdlib.org/uc/item/59n1g12w.

[36] J. R. R. A. Martins and A. B. Lambe. Multidisciplinary design optimization: a survey of architectures. *AIAA journal*, 51(9):2049–2075, 2013.

[37] S. Kodiyalam, R. J. Yang, L. Gu, and C. H. Tho. Multidisciplinary design optimization of a vehicle system in a scalable, high performance computing environment. *Structural and Multidisciplinary Optimization*, 26(3-4):256–263, 2004.

[38] C. C. Long, A. L. Marsden, and Y. Bazilevs. Shape optimization of pulsatile ventricular assist devices using FSI to minimize thrombotic risk. *Computational Mechanics*, 54(4):921–932, 2014.

[39] G. K. W. Kenway and J. R. R. A. Martins. Multipoint high-fidelity aerostructural optimization of a transport aircraft configuration. *Journal of Aircraft*, 51:144–160, 2014.

[40] T. Ashuri, M. B. Zaaijer, J. R. R. A. Martins, G J. W. van Bussel, and G. A. M. van Kuik. Multidisciplinary design optimization of offshore wind turbines for minimum levelized cost of energy. *Renewable Energy*, 68:893–905, 2014.

[41] A. L. Marsden. Optimization in cardiovascular modeling. *Annual Review of Fluid Mechanics*, 46:519–546, 2014.

[42] Mathworks. Optimization toolbox user's guide, 2016.

[43] Dakota. https://dakota.sandia.gov/. Accessed 27 May 2016.

[44] ANSYS Workbench Platform. http://www.ansys.com/Products/Platform. Accessed 27 May 2016.

[45] L. Solano and P. Brunet. Constructive constraint-based model for parametric cad systems. *Computer-Aided Design*, 26(8):614–621, 1994.

[46] X. Chen and C. M. Hoffmann. On editability of feature-based design. *Computer-aided design*, 27(12):905–914, 1995.

[47] C. M. Hoffmann. Constraint-based computer-aided design. *Journal of Computing and Information Science in Engineering*, 5(3):182–187, 2005.

[48] J. R. Rossignac, P. Borrel, and L. R. Nackman. Interactive design with sequences of parameterized transformations. In V. Akman, P. J. W. ten Hagen, and P. J. Veerkamp, editors, *Intelligent CAD Systems II: Implementational Issues*, pages 93–125. Springer-Verlag, 1989.

[49] J. Kiendl, K.-U. Bletzinger, J. Linhard, and R. Wüchner. Isogeometric shell analysis with Kirchhoff–Love elements. *Computer Methods in Applied Mechanics and Engineering*, 198: 3902–3914, 2009.

[50] J. Kiendl, Y. Bazilevs, M.-C. Hsu, R. Wüchner, and K.-U. Bletzinger. The bending strip method for isogeometric analysis of Kirchhoff–Love shell structures comprised of multiple patches. *Computer Methods in Applied Mechanics and Engineering*, 199:2403–2416, 2010.

[51] Y. Bazilevs, M.-C. Hsu, J. Kiendl, R. Wüchner, and K.-U. Bletzinger. 3D simulation of wind turbine rotors at full scale. Part II: Fluid–structure interaction modeling with composite blades. *International Journal for Numerical Methods in Fluids*, 65:236–253, 2011.

[52] A. Korobenko, M.-C. Hsu, I. Akkerman, J. Tippmann, and Y. Bazilevs. Structural mechanics modeling and FSI simulation of wind turbines. *Mathematical Models and Methods in Applied Sciences*, 23(2):249–272, 2013.

[53] Y. Bazilevs, A. Korobenko, X. Deng, and J. Yan. Novel structural modeling and mesh moving techniques for advanced fluid-structure interaction simulation of wind turbines. *International*

Journal for Numerical Methods in Engineering, 102(3-4):766–783, 2014.

[54] Y. Bazilevs, K. Takizawa, and T. E. Tezduyar. *Computational Fluid–Structure Interaction: Methods and Applications*. John Wiley & Sons, Chichester, 2013.

[55] D. J. Benson, Y. Bazilevs, M.-C. Hsu, and T. J. R. Hughes. Isogeometric shell analysis: The Reissner–Mindlin shell. *Computer Methods in Applied Mechanics and Engineering*, 199: 276–289, 2010.

[56] D. J. Benson, S. Hartmann, Y. Bazilevs, M.-C. Hsu, and T. J. R. Hughes. Blended isogeometric shells. *Computer Methods in Applied Mechanics and Engineering*, 255:133–146, 2013.

[57] R. N. Simpson, S. P. A. Bordas, J. Trevelyan, and T. Rabczuk. A two-dimensional Isogeometric Boundary Element Method for elastostatic analysis. *Computer Methods in Applied Mechanics and Engineering*, 209–212:87–100, 2012.

[58] A. I. Ginnis, K. V. Kostas, C. G. Politis, P. D. Kaklis, K A Belibassakis, Th. P. Gerostathis, M. A. Scott, and T. J. R. Hughes. Isogeometric boundary-element analysis for the wave-resistance problem using T-splines. *Computer Methods in Applied Mechanics and Engineering*, 279:425–439, 2014.

[59] M.-C. Hsu, D. Kamensky, F. Xu, J. Kiendl, C. Wang, M. C. H. Wu, J. Mineroff, A. Reali, Y. Bazilevs, and M. S. Sacks. Dynamic and fluid–structure interaction simulations of bioprosthetic heart valves using parametric design with T-splines and Fung-type material models. *Computational Mechanics*, 55:1211–1225, 2015.

[60] M.-C. Hsu, C. Wang, F. Xu, A. J. Herrema, and A. Krishnamurthy. Direct immersogeometric fluid flow analysis using B-rep CAD models. *Computer Aided Geometric Design*, 43:143–158, 2016.

[61] D. J. Benson, Y. Bazilevs, M.-C. Hsu, and T. J. R. Hughes. A large deformation, rotation-free, isogeometric shell. *Computer Methods in Applied Mechanics and Engineering*, 200: 1367–1378, 2011.

[62] J. Kiendl. *Isogeometric Analysis and Shape Optimal Design of Shell Structures*. PhD thesis, Lehrstuhl für Statik, Technische Universität München, 2011.

[63] L. Leifsson and S. Koziel. Variable-fidelity aerodynamic shape optimization. In S. Koziel and X.-S. Yang, editors, *Computational Optimization, Methods and Algorithms*, chapter 9, pages 179–210. Springer-Verlag Berlin Heidelberg, 2011.

[64] V. Torczon. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*,

7:1–25, 1997.

[65] J. Jonkman, S. Butterfield, W. Musial, and G. Scott. Definition of a 5-MW reference wind turbine for offshore system development. Technical Report NREL/TP-500-38060, National Renewable Energy Laboratory, Golden, CO, 2009.

[66] J. M. Jonkman and M. L. Buhl Jr. FAST user's guide. Technical Report NREL/EL-500-38230, National Renewable Energy Laboratory, Golden, CO, 2005.

[67] B. R. Resor. Definition of a 5MW/61.5m wind turbine blade reference model. Technical Report SAND2013-2569, Sandia National Laboratories, Albuquerque, NM, 2013.

[68] B. Snyder and M. J. Kaiser. Ecological and economic cost-benefit analysis of offshore wind energy. *Renewable Energy*, 34:1567–1578, 2009.

[69] C. Moné, T. Stehly, B. Maples, and E. Settle. 2014 cost of wind energy review. Technical Report NREL/TP-6A20-64281, National Renewable Energy Laboratory, Golden, CO, 2015.

[70] M. Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics*, 9(3):245–261, 1990.

[71] W. Martin and E. Cohen. Representation and extraction of volumetric attributes using trivariate splines: A mathematical framework. In *Proceedings of the Sixth ACM Symposium on Solid Modeling and Applications*, pages 234–240, 2001.

[72] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

[73] P. Cignoni, L. De Floriani, C. Montani, E. Puppo, and R. Scopigno. Multiresolution modeling and visualization of volume data based on simplicial complexes. In *Proceedings of the 1994 Symposium on Volume Visualization*, pages 19–26, 1994.

[74] A. Knoll, I. Wald, S. Parker, and C. Hansen. Interactive isosurface ray tracing of large octree volumes. In *2006 IEEE Symposium on Interactive Ray Tracing*, pages 115–124, 2006.

[75] B. Nelson and R. M. Kirby. Ray-tracing polymorphic multidomain spectral/hp elements for isosurface rendering. *IEEE Transactions on Visualization and Computer Graphics*, 12(1): 114–125, 2006.

[76] T. Martin, E. Cohen, and R. M. Kirby. Direct isosurface visualization of hex-based high-order geometry and attribute representations. *IEEE Transactions on Visualization and Computer Graphics*, 18(5):753–766, 2012.

[77] L. Piegl and W. Tiller. *The NURBS Book (Monographs in Visual Communication), 2nd ed.* Springer-Verlag, New York, 1997.

[78] RhinoCommon. http://developer.rhino3d.com/guides/#rhinocommon. Accessed 27 May 2016.

[79] J. F. Manwell, J. G. McGowan, and A. L. Rogers. *Wind Energy Explained: Theory, Design and Application, 2nd ed.* John Wiley & Sons, Chichester, 2009.

[80] IRENA. Renewable energy technologies: Cost analysis series, wind power. Technical report, International Renewable Energy Agency, 2012.