Lagrange extraction and projection for spline basis functions: a direct link between isogeometric and standard nodal finite element formulations

Dominik Schillinger\textsuperscript{a}, Praneeth K. Ruthala\textsuperscript{b}, Lam H. Nguyen\textsuperscript{a}

\textsuperscript{a}Department of Civil, Environmental and Geo-Engineering, University of Minnesota, USA
\textsuperscript{b}School of Mathematics, University of Minnesota, USA

Abstract

We introduce Lagrange extraction and projection that link a $C^0$ nodal basis with a smooth spline basis. Our technology is equivalent to Bézier extraction, but offers several algorithmic simplifications based on the interpolatory property of nodal basis functions. The Lagrange extraction operator can be constructed by simply evaluating spline basis functions at nodal points and eliminates the need for introducing Bernstein polynomials as new shape functions. The Lagrange projection operator is defined as the inverse of the Lagrange extraction operator and directly relates function values at nodal points to element-level spline coefficients of a local interpolant. For geometries based on polynomial splines, our technology allows the implementation of isogeometric analysis in standard nodal finite element codes with very simple algorithms and minimal intrusion. Therefore, this paper is particularly addressed to computational analysts familiar with standard finite elements who are looking for a gentle hands-on first approach to isogeometric analysis.

Keywords: Isogeometric analysis, Bézier extraction, Lagrange extraction, Local projection, Gentle first approach to IGA

*Corresponding author;
Department of Civil, Environmental, and Geo-Engineering, University of Minnesota, 500 Pillsbury Drive S.E., Minneapolis, MN 55455-0116, USA; Phone: +1 612 624-0063; Fax: +1 612 626-7750; E-mail: dominik@umn.edu
## Contents

1 Introduction 3

2 Lagrange extraction of smooth splines 5
   2.1 Lagrange decomposition ........................................ 5
   2.2 The Lagrange extraction operator ............................. 8
   2.3 Rational Lagrange basis functions and control points ...... 11
   2.4 Implementing Lagrange extraction in a standard nodal FEA code ........ 13
      2.4.1 Pre- and postprocessing .................................... 13
      2.4.2 Element-level subroutines ................................. 14
      2.4.3 A gentle introduction to IGA ............................... 14

3 Lagrange projection onto smooth splines 15
   3.1 Global $L^2$ projection ........................................ 15
   3.2 Collocation-type local $L^2$ projection and the Lagrange projection operator 16
   3.3 Projection onto a rational basis ............................... 19

4 Numerical examples 19
   4.1 Lagrange extraction for a 2D NURBS example ................ 20
   4.2 Lagrange projection for imposing Dirichlet boundary functions .... 21
   4.3 Lagrange projection for spline interpolation over the complete mesh ...... 22

5 Summary and conclusions 23

Appendix A Lagrange extraction: A gentle first approach to IGA 25
   Appendix A.1 Generating element Lagrange extraction operators ........ 25
   Appendix A.2 Generating element Lagrange control points without CAD ... 27
1. Introduction

Isogeometric analysis (IGA) [1, 2] is an isoparametric finite element method that uses smooth and higher-order spline basis functions for the representation of the geometry and the approximation of solutions fields. Splines are ubiquitous in computer aided geometric design (CAD) to represent geometric objects. The original objective of IGA has been a change of paradigm in finite element analysis (FEA): Instead of transferring the CAD model into a $C^0$ finite element mesh, the finite element method uses the spline basis functions of the CAD model directly for analysis, which opens the door for a better integration of FEA and CAD. In addition, IGA offers further important advantages. It exhibits increased accuracy and robustness on a per-degree-of-freedom basis [3–6]. Approximations of derivative fields, e.g. stresses in elasticity, are smooth and their degree can be adjusted to what is required by the variational formulation [7–9]. Unlike $C^0$ basis functions, the higher modes of spline basis functions in IGA do not diverge with increasing degree, but achieve almost spectral accuracy that improves with degree [10–12]. These properties also recently motivated the use of splines beyond IGA, e.g., in immersed-boundary-type methods [13–18].

Beyond their favorable approximation properties, the technical aspects of smooth spline basis functions raise the question of their efficient implementation. The concept of Bézier extraction has proved a milestone in this regard and has revolutionized the way IGA capabilities are implemented. The structure of Bézier elements allows the integration of IGA in existing standard FEA codes, which are typically built around element subroutines. Bézier extraction was initially introduced for non-uniform rational B-splines (NURBS) [19] and later generalized to T-splines [20, 21], hierarchical B-splines [22], hierarchical T-splines [23], and LR B-splines [24]. Bézier extraction is based on the representation of smooth basis functions in terms of $C^0$ Bernstein polynomials that are defined over each element of the spline discretization. A Bézier extraction operator can be established for each element that casts the relation between the vector of smooth basis functions and the vector of Bernstein polynomials concisely in matrix form. The extraction operator in conjunction with a Bernstein shape function subroutine allows the implementation of isogeometric analysis as an element-level technology, where all significant changes can be implemented in the formation subroutine of a standard FEA code.

The concept of Bézier extraction has been recently extended to Bézier projection by Derek Thomas and co-workers at BYU [25]. Bézier projection is a technique for obtaining an approximate $L^2$ projection of a function onto the smooth spline basis that uses only local element-level operations. This significantly decreases computational cost as compared to global projection (which requires the formation and solution of a global system of equations), but still converges optimally and leads to results that are virtually indistinguishable from global $L^2$ projection. We anticipate that local projection techniques will immediately open the door for significant progress in the design of efficient isogeometric finite element technologies. For instance, they enable fast $h$-, $p$-, $k$- and $h$-$p$-$k$-refinement and -coarsening algorithms in NURBS, T-spline and other spline discretizations [25]. Another example of significant interest is the mitigation of locking phenomena by efficiently projecting fields onto approximations of lower order and continuity [26–30].
In this paper we introduce the concept of Lagrange extraction and projection. It is equivalent to Bézier extraction and projection, but instead of linking smooth spline basis functions to $C^0$ Bernstein polynomials, it establishes a direct link between splines and nodal finite element basis functions. Although a replication of Bézier extraction to some extent, Lagrange extraction offers algorithmic simplifications that could lower potential hurdles to adopting isogeometric capabilities in standard FEA codes. They are based on the interpolatory property of Lagrange basis functions at element nodes, which can be exploited to establish compact and very simple algorithms that reduce cost and algorithmic complexity with respect to corresponding Bézier operations. The most important example is the straightforward construction of the Lagrange extraction operator, which maps a nodal Lagrange polynomial basis to a smooth spline basis. It can be directly assembled in each element by simply evaluating spline basis functions at element nodes. This can be implemented in just a few lines of code (see simple MATLAB examples in the Appendix) without additional spline-specific algorithms (e.g., knot insertion) that are required for the construction of the Bézier extraction operator. Another example is our version of local Lagrange projection, which can be interpreted as a localized projection operation related to interpolation of a smooth spline function by a nodal Lagrange basis [25, 31, 32]. We use a collocation-type nodal formation and assembly concept [33–35] that defines the Lagrange projection operator as the inverse of the extraction operator. This results in a local quadrature-free procedure that is able to directly relate Lagrange coefficients to element-level spline coefficients by a single matrix-vector product. The Lagrange projection operator does not have to be computed and stored separately, but can be obtained by local matrix inversion if needed.

The algorithmic simplifications of Lagrange extraction can be an important advantage for IGA beginners who are interested in testing the benefits of IGA for their specific applications. Our technology opens the door for running simple prototype IGA calculations in standard FEA codes with minimal intrusion. Lagrange extraction eliminates the need for introducing Bernstein polynomials as new shape functions, so that no spline functions have to be implemented at any point of the element subroutine. Since the formation of the element stiffness matrix can still be accomplished with nodal basis functions, all formation procedures at the element level remain unchanged. The only modification that is required is the adjustment of the Jacobian, which has to accommodate rational basis functions for NURBS and T-splines. If we restrict ourselves to polynomial splines, e.g. B-splines, finite element subroutines of standard FEA codes can remain completely untouched at the element level, and the only modifications necessary are additional matrix-matrix or matrix-vector products that transform local arrays before assembly to global arrays.

The paper is organized as follows: Section 2 deals with Lagrange extraction of a spline basis. In particular, we will discuss the construction of the Lagrange extraction operator, the modification of control points and the handling of weights, when the spline basis is rational. Section 3 focuses on Lagrange projection onto a smooth spline basis. We will show that the Lagrange projection operator directly relates the nodal values of the function to be projected to element-level spline coefficients. For optimal global accuracy, we will use a simple weighted averaging scheme [25] when assembling the local element-level spline coefficients into the global vector of spline coefficients. Section 4 presents numerical exam-
amples, computed with nodal finite elements and Lagrange extraction. We will compare the accuracy per degree of freedom in IGA vs. standard FEA for a 2D elasticity example and demonstrate that local Lagrange projection converges optimally and achieves results that are virtually indistinguishable from global $L^2$ projection.

2. Lagrange extraction of smooth splines

We will first discuss the decomposition of a smooth piecewise polynomial B-spline basis into a $C^0$ nodal Lagrange basis. The construction of the Lagrange extraction operator is solely based on the evaluation of splines at element nodes, which can be implemented in a few lines of code. Corresponding MATLAB implementations for 1D, 2D and 3D discretizations are provided in the Appendix. We will also extend Lagrange extraction to rational spline basis functions of non-uniform rational B-splines (NURBS).

2.1. Lagrange decomposition

A B-spline curve of polynomial degree $p$ is defined by a linear combination of $n$ B-spline basis functions. We define the set of B-spline basis functions as $\mathbf{N}(r) = \{N_{a,p}(r)\}_{a=1}^{n}$. The corresponding set of vector-valued control points is denoted as $\mathbf{P} = \{P_a\}_{a=1}^{n}$, where each $P_a \in \mathbb{R}^d$, $d$ being the number of spatial dimensions. $\mathbf{P}$ can be represented as a $n \times d$ matrix

$$
\mathbf{P} = \begin{bmatrix}
P_1^1 & P_2^1 & \cdots & P_d^1 \\
P_1^2 & P_2^2 & \cdots & P_d^2 \\
\vdots & \vdots & \ddots & \vdots \\
P_n^1 & P_1^n & \cdots & P_d^n
\end{bmatrix}
$$

(1)

The B-spline curve $S$ can then be written as

$$
S(r) = \sum_{a=1}^{n} P_a \cdot N_{a,p}(r) = \mathbf{P}^T \mathbf{N}(r)
$$

(2)

where $r \in \mathbb{R}$ is a non-decreasing parametric coordinate. An example of a two-dimensional cubic B-spline curve along with the corresponding B-spline basis functions and control points is shown in Fig. 1. The B-spline basis generated from an open knot vector [2, 36] is $C^2$-continuous in the entire domain. The parameter space over which B-spline basis functions are defined can be split into knot spans $\hat{\Omega}^e$ (see Fig. 1 for the example basis). We denote knot spans as elements in the following.

We now focus on a single element $e$ and introduce a new parametric coordinate $\xi \in [-1, 1]$ for that element. Element quantities are denoted by superscript $e$. The $p+1$ B-spline basis functions with support over that element form a linearly independent and complete polynomial basis up to degree $p$. It is possible to represent the B-spline basis functions over that element by a linear combination of the basis functions of any other polynomial basis that is linearly independent and complete up to polynomial degree $p$. The standard element
nodal basis satisfies these two properties. For each B-spline function, we can therefore write
\[ N^e_{a,p}(\xi) = \sum_{b=1}^{p+1} \alpha_{ab} L_{b,p}(\xi) \]  
(3)
or in more compact form
\[ \mathbf{N}^e(\xi) = \mathbf{D}^e \mathbf{L}(\xi) \]  
(4)
where the set of standard Lagrange basis functions is defined as \( \mathbf{L}(\xi) = \{L_{a,p}(\xi)\}_{a=1}^{p+1} \) with associated nodal points \( \hat{\xi}_a \) on the element \( e \). Note that element nodal basis functions do not require an index \( e \), since they are identical in each element. This is not the case for B-splines, and each element set of functions is therefore indexed. The transformation matrix \( \mathbf{D}^e \) maps the nodal Lagrange basis onto the B-spline basis. It contains the coefficients \( \alpha_{ab} \) that represent the fraction of each nodal basis function in a B-spline basis function and is called the Lagrange extraction operator for the given element.

We can insert the extraction (4) back into (2) to obtain the representation of the B-spline curve in terms of nodal Lagrange basis functions. For the curve segment defined over an
(a) Lagrange curve and Lagrange control points.

(b) Corresponding cubic nodal basis.

Figure 2: Smooth $C^2$-continuous curve represented by a nodal Lagrange basis.

Combining the element-level analysis for our example curve, we plot the Lagrange control points and the Lagrange basis functions over each element in Figs. 2a and 2b. We observe that the Lagrange representation leads to exactly the same curve than the B-spline representation of Fig. 1a. In particular, the curve represented by Lagrange basis functions and Lagrange control points is still higher-order continuous between curve segments, although the basis functions are $C^0$ continuous at element boundaries. The Lagrange control points are interpolatory and represent element nodes in the standard FEA sense.
2.2. The Lagrange extraction operator

In the next step, we show how the Lagrange extraction operator can be computed. The central property of a Lagrange basis is its interpolatory property at the nodal points $\hat{\xi}_a$ [37]. This means that for each basis function $L_{a,p}$ there exists a node $\hat{\xi}_a$, where the function itself is one and all other nodal basis functions are zero. This can be summarized as

$$L_{a,p}(\hat{\xi}_b) = \begin{cases} 1.0 & \text{if } a = b \\ 0.0 & \text{if } a \neq b \end{cases}$$  \hspace{1cm} (7)

The interpolatory property at element nodes can be easily observed in Fig. 2b. Using (7) on each element, an interpolation $\bar{u}(\xi)$ of a function $u(\xi)$ with the nodal basis functions $L_{a,p}$ can be obtained in the following form (Lagrange interpolation) [37]

$$\bar{u}(\xi) = \sum_{a=1}^{p+1} L_{a,p} u(\hat{\xi}_a)$$  \hspace{1cm} (8)

where coefficients $u(\hat{\xi}_a)$ are simply evaluations of that function at the corresponding nodal points. If $u(\xi)$ is a polynomial of degree $p$ or smaller, $\bar{u}(\xi) = u(\xi)$.

From (7) and (8), it is now straightforward to see that we can find the coefficients in the $a^{th}$ row of the element Lagrange extraction operator (4) by simply evaluating the corresponding B-spline basis function $N_{a,p}^e$ at all element nodes $\{\hat{\xi}_b\}_{b=1}^{p+1}$. The complete Lagrange extraction operator for a one-dimensional element thus reads

$$D_{ab}^e = N_{a,p}^e(\hat{\xi}_b), \quad \{a,b\} = 1,\ldots,p+1$$  \hspace{1cm} (9)

or in matrix form

$$D^e = \begin{bmatrix} N_{1,p}^e(\hat{\xi}_1) & N_{1,p}^e(\hat{\xi}_2) & \cdots & N_{1,p}^e(\hat{\xi}_{p+1}) \\ N_{2,p}^e(\hat{\xi}_1) & N_{2,p}^e(\hat{\xi}_2) & \cdots & N_{2,p}^e(\hat{\xi}_{p+1}) \\ \vdots & \vdots & \ddots & \vdots \\ N_{p+1,p}^e(\hat{\xi}_1) & N_{p+1,p}^e(\hat{\xi}_2) & \cdots & N_{p+1,p}^e(\hat{\xi}_{p+1}) \end{bmatrix}$$  \hspace{1cm} (10)

We note that there exists a global Lagrange extraction operator, which relates nodal basis functions and B-spline basis functions defined over the complete mesh. However, in the context of finite element data structures, we are interested in element-level computations using element extraction operations, so that the global operator is never constructed. The concept of Lagrange extraction illustrated here for the one-dimensional case directly generalizes for two and three-dimensional elements.

A multi-dimensional Lagrange extraction operator is constructed by simply evaluating tensor-product B-spline functions at nodal points. We note that Lagrange extraction as presented here requires a full tensor-product Lagrange basis to guarantee completeness with respect to the tensor-product B-spline basis. We provide MATLAB codes for the generation of element Lagrange extraction operators for single patch tensor-product spline discretizations.
in one, two and three dimensions in the Appendix. For multi-dimensional spline elements with anisotropic polynomial degree, that is \( p \) is different in each parametric direction, the corresponding Lagrange extraction operator can be constructed for a nodal Lagrange basis that is constructed with equivalent polynomial degree \( p \) in each parametric direction.

In CAD, geometric objects are usually transferred in the form of Bézier element representations, which can be exported from any CAD software. Bézier elements are based on \( C^0 \) Bernstein polynomials \( B(\xi) = \{B_{a,p}(\xi)\}_{a=1}^{p+1} \) \([19, 38]\) on each element. Since the set of basis function is identical for each element, they do not need an index \( e \). For our example curve, the corresponding Bézier control points and element basis functions are shown in Figs. 3a and 3b. Leveraging again the arguments of completeness and linear independence, we know that there exists a Bézier extraction operator \( C \) that maps a piecewise Bernstein polynomial basis onto a smooth B-spline basis on each element. The construction of the Bézier extraction operator and associated Bézier control points can be achieved by Bézier decomposition \([19]\), which is largely based on knot insertion \([19, 20, 36]\).

It should be mentioned here that the Lagrange extraction operator could be obtained for each element of a Bézier representation by evaluating Bernstein polynomials at nodal points instead of B-splines in (9) and (10). We call the resulting transformation matrix the
Smooth B-splines

Figure 4: Illustration of the extraction operators and their inverse for the transformation of B-spline, Lagrange and Bernstein basis functions on an element level. The B-spline element refers to the second element of our example curve (see Fig. 1).

Lagrange-Bernstein operator, which has the following form

\[ D_{lb} = \begin{bmatrix}
B_{1,p}(\hat{\xi}_1) & B_{1,p}(\hat{\xi}_2) & \ldots & B_{1,p}(\hat{\xi}_{p+1}) \\
B_{2,p}(\hat{\xi}_1) & B_{2,p}(\hat{\xi}_2) & \ldots & B_{1,p}(\hat{\xi}_{p+1}) \\
\vdots & \vdots & \ddots & \vdots \\
B_{p+1,p}(\hat{\xi}_1) & B_{p+1,p}(\hat{\xi}_2) & \ldots & B_{p+1,p}(\hat{\xi}_{p+1})
\end{bmatrix} \]  

The Lagrange-Bernstein operator is identical for each element. We can extend a Bézier extraction operator by multiplying with the Lagrange-Bernstein operator as follows

\[ D^e = C^e D_{lb} \]  

and obtain the Lagrange extraction operator (10). The corresponding Lagrange control points \( P^l \) can be obtained from the Bézier control points \( P^b \) as

\[ P^l = (D^{lb})^T P^b \]  

Figure 4 summarizes the element-level transformation relations between B-spline basis functions, Lagrange basis functions and Bernstein polynomials. It illustrates that for each element each set of basis functions can be obtained from each other set of basis functions by a sequence of transformation operators. Figure 5 summarizes the element-level transformation between corresponding B-spline, Lagrange and Bézier control points for the representation of the same smooth curve. We note that the transformation matrices are the same as for the
corresponding basis functions, but need to be transposed. We emphasize that relations (11), (12) and (13) complete the diagram, but are of restricted practical value. In case the Bézier extraction operator is known, we expect that one would always use element-level Bernstein basis functions instead of performing an additional transformation. We therefore attenuate the link between Lagrange and Bernstein basis functions and control points in Figs. 4 and 5. There exist inverse operators for the Lagrange, Bézier and Lagrange-Bernstein operators that establish a connection between the corresponding basis functions and control points in the opposite direction. The inverse of the Lagrange extraction operator is called the Lagrange projection operator and will be discussed in the next section.

2.3. Rational Lagrange basis functions and control points

In general, spline discretizations consist of rational basis functions, which affects the computation of Lagrange basis functions and control points. In the following, we briefly illustrate this for the case of non-uniform rational B-splines (NURBS) in one dimension. The procedure described in the following is equivalent to representing NURBS by $C^0$ Bézier elements, and we closely follow the presentation of Borden [19].

We define the set of NURBS basis functions $R^e(\xi) = \{R^{e}_{a,p}(\xi)\}_{a=1}^{p+1}$ with associated control points $P^e = \{P_a\}_{a=1}^{p+1}$ on each element $e$ as

$$R^{e}_{a,p}(\xi) = \frac{w_a N^{e}_{a,p}(\xi)}{W(\xi)} \quad (14)$$

where $w_a$ is the weight corresponding to the $a^{th}$ basis function with support on the element.
The weight function $W$ is defined as

$$W(\xi) = \sum_{b=1}^{p+1} w_b N_{b,p}^e(\xi)$$  \hspace{1cm} (15)

We now define $W$ to be the diagonal matrix of weights

$$W = \begin{bmatrix} w_1 & & \cdots & \ & w_{p+1} \\ \end{bmatrix}$$  \hspace{1cm} (16)

Dropping subscripts $p$ and $e$ for clarity, we can write (14) in matrix form as

$$R(\xi) = \frac{1}{W(\xi)} W N(\xi)$$  \hspace{1cm} (17)

We now can express each segment of a spline curve parameterized by NURBS basis functions in terms of the nodal Lagrange basis as

$$S(\xi) = P^T R(\xi) = \frac{1}{W(\xi)} P^T W N(\xi)$$

$$= \frac{1}{W(\xi)} P^T W D L(\xi) = \frac{1}{W(\xi)} (D^T W P)^T L(\xi)$$  \hspace{1cm} (18)

Grouping the weights into a vector $\mathbf{w} = \{w_a\}_{a=1}^{p+1}$ we can rewrite the weight function, $W(\xi)$, in terms of the nodal Lagrange basis as

$$W(\xi) = \mathbf{w}^T N(\xi) = \mathbf{w}^T D L(\xi) = (D^T \mathbf{w})^T L(\xi) = (\mathbf{w}^l)^T L(\xi) = W^l(\xi)$$  \hspace{1cm} (19)

where $\mathbf{w}^l = D^T \mathbf{w}$ are the weights associated with the Lagrange basis functions. To compute the Lagrange control points, $P^l$, we first define $W^l$ to be the diagonal matrix

$$W^l = \begin{bmatrix} w^l_1 & & \cdots & \ & w^l_{p+1} \\ \end{bmatrix}$$  \hspace{1cm} (20)

The Lagrange control points are now computed as

$$P^l = (W^l)^{-1} D^T W P$$  \hspace{1cm} (21)

This equation can be interpreted as mapping the original NURBS control points into a projective space [19, 36], applying the Lagrange extraction operator to compute the control
points of the projected Lagrange elements and then mapping these control points back from projective space. Multiplying (21) by \( W_l \) we find the relationship

\[
\mathbf{D}^T \mathbf{W} \mathbf{P} = \mathbf{W}^l \mathbf{P}^l
\]  

(22)

To obtain the final Lagrange representation of a NURBS curve we substitute (20) and (22) into (18) to get

\[
S(\xi) = \mathbf{P}^T \mathbf{R}(\xi) = \frac{1}{W(\xi)} \mathbf{P}^T \mathbf{W} \mathbf{N}(\xi)
\]

\[
= \frac{1}{W^l(\xi)} (\mathbf{P}^l)^T \mathbf{W}^l \mathbf{L}(\xi) = \sum_{a=1}^{n} \frac{w^a_l L_{a,p}(\xi)}{W^l(\xi)} \mathbf{P}^l_a
\]  

(23)

Thus, we have shown that each segment of a smooth NURBS curve can be written equivalently in terms of a set of \( C^0 \) nodal Lagrange elements. We emphasize again that the NURBS curve represented by rational Lagrange basis functions is still smooth between curve segments, although the nodal basis functions are only \( C^0 \) continuous at element boundaries.

We note that the concept of rational Lagrange basis functions introduced here allows the straightforward extension of Lagrange extraction to other rational spline discretizations, for example T-splines [39, 40]. However, rational Lagrange basis functions require special algorithms to compute derivatives and the Jacobian mapping. This requires significant adjustment and addition of capabilities within element subroutines of standard FEA codes, which are based on polynomial basis functions only. Therefore, the advantage of Lagrange extraction in terms of minimal intrusion, which holds for polynomial splines, is mitigated for rational basis functions.

2.4. Implementing Lagrange extraction in a standard nodal FEA code

We assume that the geometric object to be analyzed is designed with a CAD tool. The exact geometry in terms of Lagrange elements, the Lagrange extraction operator, Lagrange control points, and Lagrange weights need to be generated from the data that can be exported from the CAD tool. The latter usually consist of degree-of-freedom information for the smooth discretization (degree-of-freedom index per basis function, connectivity information) and information on how to construct the spline basis functions, i.e. knot vectors, weights and spline control points. We anticipate that in most cases it will be the best option to implement a small preprocessing tool outside of the FEA code that can generate the Lagrange extraction operator using (9) and (10). The operations left to be implemented inside the FEA tool are described in the following.

2.4.1. Pre- and postprocessing

We begin by reading in the Lagrange extraction operator, the Lagrange control points (possibly with weights), and the degree-of-freedom information of the spline discretization. We store this information for each element separately in the element data structure. We use the
degree-of-freedom information of the spline discretization to allocate global arrays, i.e. the
global stiffness matrix and load vector.

After solution of the global system of equations, we can use existing subroutines of
standard nodal FEA codes to evaluate solution fields at visualization points. In each element
we can apply relation (6) to transform solution coefficients of the spline basis to solution
coefficients of the nodal Lagrange basis. If derivative fields need to be visualized and the
basis is rational, we need to adjust the computation of the Jacobian matrix.

2.4.2. Element-level subroutines

If we use geometric objects parameterized by a B-spline mesh (no rational basis functions),
the element subroutines of a standard nodal finite element code do not need to be touched.
This can be a fundamental advantage in a number of situations, for example, when we do
not have access to the complete code or when subroutines based on nodal basis functions are
highly optimized and should not be touched. In the case of B-splines, we can hand over the
Lagrange control points as the standard nodal coordinates to the element subroutine. After
computing the standard nodal arrays for each element, the only modifications necessary are
additional matrix-matrix and matrix-vector products that transform local arrays from the
$C^0$ Lagrange to the smooth spline setting before assembly to corresponding global arrays.
These operations read for each element

$$ k_{iga}^e = (D^e)k_{fea}^e(D^e)^T $$
$$ f_{iga}^e = (D^e)f_{fea}^e $$

where $k_{iga}^e$ and $k_{fea}^e$ are the element stiffness matrices, and $f_{iga}^e$ and $f_{fea}^e$ are the element load
vectors that correspond to the smooth spline case and the $C^0$ Lagrange case, respectively.
We emphasize again that the local arrays $k_{iga}^e$ and $f_{iga}^e$ are exactly the same as the local arrays computed directly with the smooth spline basis functions and can be assembled into
the corresponding global arrays. In Section 4.1, we review the element local stiffness matrix
and the element load vector in the context of Lagrange extraction for the example of 2D
elasticity in more detail.

If we use rational splines, we additionally need to adjust the computation of basis function
derivatives, the Jacobian matrix and its determinant in each element, since Lagrange weights
and the weight function need to be incorporated. These operations are not part of standard
FEA codes that are restricted to polynomial basis functions. For more details on computing
derivatives of rational basis functions, we refer to Cottrell et al. [2] and Borden et al. [19].

2.4.3. A gentle introduction to IGA

The procedures described here are particularly well suited for IGA beginners who are
at a preliminary stage and want to run prototype computations to rapidly test the effect
of smooth basis functions for their application. Lagrange extraction opens the door for a
gentle first approach to isogeometric analysis that enables computational analysts familiar
with standard FEA to implement IGA capabilities with minimal intrusion in a standard FEA
code. In the Appendix, we provide simple MATLAB codes that illustrate how to construct
the Lagrange extraction operator. We also discuss how to generate Lagrange control points without using a CAD tool for simple geometries that can be used for first IGA computations.

3. Lagrange projection onto smooth splines

In finite element analysis, the interpolation of a given function with the finite element basis functions is an important operation that is required in many situations. Examples include curve and surface fitting, imposition of Dirichlet boundary conditions, mesh adaptivity, or multi-level and multi-scale procedures, just to name a few. Using standard nodal finite elements, we can exploit the interpolatory property of the Lagrange basis and find the coefficients of the interpolant by simply evaluating the target function at nodal points (Lagrange interpolation) [37]. However, splines are not interpolatory, and function interpolation therefore has to be done by projection. Unfortunately, projection is a global operation that is computationally expensive due to the assembly and solution of a global system of equations. To reduce cost, we are interested in efficient local projection schemes that can operate locally on each element and lead to a interpolant that delivers the same order of accuracy as global projection. In this section, we will establish the Lagrange projection operator as the inverse of the Lagrange extraction operator and show that it directly transforms values of the target function in each element (i.e., coefficients of a Lagrange interpolation) to coefficients of a suitable spline interpolant. Lagrange projection is closely related to Bézier projection [25].

3.1. Global $L^2$ projection

$L^2$ projection is a simple projection of an arbitrary function $u \in L^2(\Omega)$ into a finite dimensional subspace $V_h \subset L^2(\Omega)$, where $\Omega \subset \mathbb{R}^d$ is a domain of dimension $d$. In summary, we are looking for the function $u_h \in V_h$ that best approximates $u$ by minimizing the error in the $L^2$ norm. Mathematically, this can be formulated by the following minimization problem: Find $u_h$ such that

$$J(u_h) := \frac{1}{2} \|u_h - u\|^2_{L^2(\Omega)} \rightarrow \min$$ (26)

The corresponding optimality condition reads

$$\int_{\Omega} (u_h - u) v_h \, d\Omega = 0 \quad \text{for all } v_h \in V_h$$ (27)

To interpolate the given function $u$ over an isogeometric mesh, we solve the variational statement (27) by discretizing the interpolant $u_h$ and the test function $v_h$ with the set of $n$ (possibly rational) spline basis functions $R = \{R_{a,p}\}_{a=1}^n$ in the usual form

$$u_h = \sum_{a=1}^{n} R_{a,p} c_a = H c \quad v_h = \sum_{a=1}^{n} R_{a,p} \delta c_a = H \delta c$$ (28)
where $H$ is the row vector of spline basis functions and $c$ and $\delta c$ are the column vectors of coefficients of the interpolant and the test function. Inserting (28) into (27) leads to the following global discrete system of equations

$$\int_{\Omega} H^T H \, d\Omega \cdot c = \int_{\Omega} H^T u \, d\Omega$$

(29)

with a global mass matrix on the left and a force vector on the right. Solving this global system we obtain the global vector of coefficients $c$ of the spline interpolant over the complete isogeometric mesh.

3.2. Collocation-type local $L^2$ projection and the Lagrange projection operator

We now want to derive an element-level projection technology for splines. Our basic idea is to transform the spline basis to a nodal basis via the Lagrange extraction operator introduced in Section 2, which opens the door for exploiting the interpolatory property of Lagrange basis functions. To this end, we restrict our attention to one element of the spline mesh, $\hat{\Omega}^e$, and assume for the moment that the functions defined there are B-spline basis functions $N^e = \{N^e_{a,p}\}_{a=1}^n$. The local element-level version of (29) reads

$$\int_{\hat{\Omega}^e} (H^e)^T H^e \, d\hat{\Omega} \cdot c^e = \int_{\hat{\Omega}^e} (H^e)^T u \, d\hat{\Omega}$$

(30)

where $c^e$ denotes the local vector of coefficients of the interpolant defined over the element $\hat{\Omega}^e$ only. For clarity of notation, we mostly drop superscript $e$ in the following.

In the next step, we use Lagrange extraction to transform the B-spline basis to a nodal basis defined by $n = (p+1)^d$ nodal points $\hat{x}_a = \{\hat{x}_a\}_{a=1}^n$ within the element. We emphasize that $x$ refers to the physical coordinates of the geometric object and its mesh, but we will see later that no geometric mapping will be required if the target function values are known at nodal points, which is the case for most applications in a finite element context. Moving the extraction operator (it is constant over the element) out of the integral we find

$$D \left( \int_{\hat{\Omega}^e} (H^l)^T H^l \, d\hat{\Omega} \right) \cdot D^T c^e = D \int_{\hat{\Omega}^e} (H^l)^T u \, d\hat{\Omega}$$

(31)

where $H^l$ denotes the row vector of nodal basis functions $L = \{L_{a,p}\}_{a=1}^n$ with support over the element.

In the next step, we want to evaluate the integrals in (31) numerically by using a quadrature rule [37]. A quadrature rule replaces an integral by the sum of weighted evaluations of the integrand $u(x)$ at $m$ discrete quadrature points $x_a$ as follows

$$\int_{\hat{\Omega}^e} u(x) \, d\hat{\Omega} = \sum_{a=1}^m u(x_a) \, \hat{\omega}_a$$

(32)

where $\hat{\omega}_a$ are the corresponding quadrature weights. Instead of choosing a known rule such
as Gauss or Newton-Cotes quadrature \cite{37}, we assume that there is a rule available whose quadrature points are identical with the nodal points \( \hat{x}_a = \{ \hat{x}_a \}_{a=1}^n \) of the Lagrange basis. With the assumed quadrature rule the Lagrange mass matrix becomes diagonal, since

\[
L_i(\hat{x}_k) L_j(\hat{x}_k) \hat{\omega}_k = \begin{cases} 
\hat{\omega}_i & \text{if } i = j = k \\
0.0 & \text{otherwise} 
\end{cases} \tag{33}
\]

Moreover, the Lagrange force vector is just the evaluation of the target function \( u \) at the nodal points, since

\[
L_i(\hat{x}_k) u(\hat{x}_k) \hat{\omega}_k = \begin{cases} 
u(\hat{x}_i) & \text{if } i = k \\
0.0 & \text{otherwise} \end{cases} \tag{34}
\]

Using (33) and (34) in (31) and bringing the element force vector to the left-hand side we obtain

\[
D^T c^e - \begin{bmatrix} u(\hat{x}_1) \\
u(\hat{x}_2) \\
\vdots \\
u(\hat{x}_m) \end{bmatrix} = 0 \tag{35}
\]

where \( m \) runs from 1 to \((p + 1)^d\). Similar strategies based on the Kronecker \( \delta \) property (33) and (34) of nodal basis functions at quadrature points have been used in many collocation-type finite element schemes, e.g. the \( C^0 \)-collocation-Galerkin method \cite{33,41–44}, the differential quadrature method \cite{45,46}, the G-NI or SEM-NI methods (Galerkin/spectral element methods with numerical integration) \cite{34,35,47}, multidomain spectral or pseudospectral elements \cite{35,48–51} or \( hp \)-FEM with Gauss-Lobatto basis functions \cite{52,53}.

It is now straightforward to see that in (35) the unknown quadrature weights cancel, when we divide each line \( a \) of the system of equations by quadrature weight \( \hat{\omega}_a \). We recall again that the Lagrange extraction operator \( D \) provides a map from the Lagrange basis to the B-spline basis for each element. Both sets of functions are linearly independent and complete, therefore the Lagrange extraction operator is full rank and invertible. It follows that in (35) the expression in brackets needs to be zero to satisfy the equation, viz.

\[
D^T c^e = 0 \tag{36}
\]

Using the invertibility of the Lagrange extraction operator we can obtain the element coef-
coefficients of the spline interpolant as

\[
\mathbf{c}^e = \mathbf{D}^{-T} \begin{bmatrix}
    u(\hat{x}_1) \\
    u(\hat{x}_2) \\
    \vdots \\
    u(\hat{x}_m)
\end{bmatrix}
\] (37)

This equation shows that for each element the inverse of the Lagrange extraction operator, which we call *Lagrange projection operator* in the following, relates simple point evaluations of the target function at the Lagrange nodes to the coefficients of a spline interpolant. If we assume that the Lagrange extraction operator is given, local spline projection is trivial from an implementation point of view. It can be achieved by a simple local inversion of a small matrix, which can be done very efficiently by standard linear algebra packages. In particular, Lagrange projection is completely quadrature-free and does not require the computation of a geometric mapping, e.g., the Jacobian. Moreover, many elements will have the same Lagrange extraction operator. It is therefore possible to compute the Lagrange projection operators for typical spline elements and simply mirror the operator from memory when the corresponding element is invoked. Local spline projection can thus be achieved with a single matrix-vector product once the function values at the physical coordinates of the nodal points are known.

Lagrange projection is an element-level technology and leads to an interpolant on each element. To obtain the global interpolant, the vector of coefficients \( \mathbf{c}^e \) from each element needs to be assembled in the vector of global coefficients \( \mathbf{c} \). However, the coefficients produced by Lagrange projection for the same spline basis function on different elements will be different. Therefore, their values must be averaged, selected, or combined in some way during assembly. We follow the procedure suggested in [25]. The global coefficient of a spline basis function is found from a weighted sum of local coefficients that are gathered from different elements, over which the spline basis function has support. For each coefficient, this averaging reads

\[
c_a = \sum_{e \in E_a} \beta_a^e c_a^e
\] (38)

where \( c_a \) and \( c_a^e \) represent the global coefficient and the local coefficient in element \( e \) of the \( a^{th} \) basis function, and the local element set \( E_a \) contains the elements in the support of basis function \( a \). Following [25] the weight \( \beta_a^e \) for basis function \( a \) on each element can be determined by the ratio between the area under the local spline function vs. the area under the global function, viz.

\[
\beta_a^e = \frac{\int_{\hat{\Omega}^e} N_a^e d\hat{\Omega}}{\sum_{e \in E_a} \int_{\hat{\Omega}^e} N_a^e d\hat{\Omega}}
\] (39)

where \( \hat{\Omega}^e \) refers to the parametric domain of the reference element. It was shown in [25] that the local averaging scheme based on (38) and (39) provides superior accuracy with respect to simpler choices of weights. For typical spline elements, the weights \( \beta_a^e \) on each element
can be precomputed and stored, which eliminates the computational cost for (39) and the associated communication between elements.

3.3. Projection onto a rational basis

Spline discretizations with rational basis functions require slight adjustments of the element-level Lagrange projection procedure introduced in Section 3.2, which we will illustrate for the example of NURBS basis functions. If we want to project on NURBS, the element-level projection (30) contains the set of NURBS functions \( \mathbf{R}^e = \{ R_{a,p}^e \}_{a=1}^n \) in the row vector \( \mathbf{H} \). Using Lagrange extraction of NURBS and moving the extraction operator out of the integral, we find

\[
D \left( \int_{\hat{\Omega}} \frac{1}{W(x)} W(H^T) \frac{1}{W(x)} W H^T \, d\hat{\Omega} \right) D^T c^e = D \int_{\hat{\Omega}} \frac{1}{W(x)} W (H^T) u \, d\hat{\Omega} \tag{40}
\]

We recall from Section 2.3 that the diagonal matrix \( W \) contains the NURBS weight of the \( a \)th basis function at the \( a \)th diagonal position, and the weight function \( W(x) \) is defined as the sum of all NURBS basis functions times their weights (refer to (16) and (19) for details). Using the interpolatory property of nodal basis functions (33) and (34) in (40) and bringing the element force vector to the left-hand side we obtain

\[
D \begin{pmatrix}
\left( \frac{\omega_1}{W(\hat{x}_1)} \right)^2 \omega_1 & \cdots & \left( \frac{\omega_m}{W(\hat{x}_m)} \right)^2 \omega_m
\end{pmatrix}
D^T c^e - \begin{pmatrix}
\left( \frac{\omega_1}{W(\hat{x}_1)} \right) \omega_1 u(\hat{x}_1) \\
\vdots \\
\left( \frac{\omega_m}{W(\hat{x}_m)} \right) \omega_m u(\hat{x}_m)
\end{pmatrix} = 0 \tag{41}
\]

Using the invertibility of the Lagrange extraction operator, we find the local spline coefficients for the rational spine basis functions from

\[
c^e = W^{-1} D^{-T} \begin{bmatrix}
W(\hat{x}_1) u(\hat{x}_1) \\
\vdots \\
W(\hat{x}_m) u(\hat{x}_m)
\end{bmatrix} \tag{42}
\]

This equation shows that the concept of Lagrange projection also works for rational spline basis functions. The two adjustments that are necessary are that we project the function \( W(x) u(x) \) and that we need to multiply with the inverse of the diagonal matrix \( W \). We emphasize that the weight function, \( W(x) = W(\xi) \), can be computed with respect to the parametric coordinates of the reference element and the inversion of a diagonal matrix is trivial. Therefore, the associated increase in computational effort is small.

4. Numerical examples

Lagrange extraction does not require numerical examples to prove its validity, as it consists entirely of identity operations. Results obtained with Lagrange extraction are thus
exactly the same as results obtained by directly using smooth basis functions. In this section, we first sketch how Lagrange extraction blends into the formulation of 2D elasticity, following up in what we presented in Section 2.4. We then briefly demonstrate the accuracy of local Lagrange projection with respect to global $L^2$ projection for (a) the imposition of Dirichlet boundary functions and (b) the projection of a known solution on a 2D NURBS mesh. We also illustrate the superior per-degree-accuracy accuracy of smooth basis functions as compared to standard FEA.

4.1. Lagrange extraction for a 2D NURBS example

We consider a two-dimensional model problem defined over a quarter of an annular section that is discretized by a single patch of NURBS of polynomial degree $p=2$ to 5. The quarter annulus is located within the positive quadrant of the Cartesian coordinate system \{x, y\}. We assume plane strain elasticity [54] with a load $b$ whose $x$- and $y$-components are

$$b_x = b_y = 16r^2 \sin(x) - 68 \sin(x) + x(8r^2 - 68) \cos(x)$$ (43)

The load is manufactured in such a way that the exact solution in terms of $x$- and $y$-displacements over the quarter cylindrical section reads

$$u_x = u_y = (r^2 - 1)(r^2 - 16) \sin(x)$$ (44)

Geometry and boundary conditions as well as the initial $9 \times 6$ mesh and the corresponding solution are illustrated in Figs. 6a and 6b, respectively. The discretized Galerkin formulation of the model problem reads

$$\sum_e \left( \int_{\hat{\Omega}^e} (B^{e})^T \mathbb{C} B^{e} d\hat{\Omega} \right) \hat{\mathbf{u}} = \sum_e \left( \int_{\hat{\Omega}^e} (H^{e})^T \mathbf{b} d\hat{\Omega} \right)$$ (45)

where $\mathbb{C}$ is the plane strain elasticity matrix, $B^{e}$ is the element strain-displacement matrix that contains derivatives of basis functions, and $\hat{\mathbf{u}}$ is the vector of global unknowns [54].

Summation of the integral terms over elements $e$ yields the global stiffness matrix $K$ and global load vector $f$. Using Lagrange extraction, the global stiffness matrix and global load vector are computed as follows

$$K = \sum_e D \int_{\hat{\Omega}^e} (B^{e,l})^T \mathbb{C} B^{e,l} d\hat{\Omega} \quad D^T$$ (46)

$$f = \sum_e D \int_{\hat{\Omega}^e} (H^{e,l})^T \mathbf{b} d\hat{\Omega}$$ (47)

where quantities computed with nodal basis functions are denoted by superscript $l$. We first compute the element stiffness matrix and element load vector on a standard element with nodal basis functions. Before assembly in the corresponding global arrays, we transform the resulting element arrays by matrix-matrix or matrix-vector products (see also (24) and (25).
in Section 2.5) using the element Lagrange extraction operator \( D \). We emphasize again that the global extraction operator does not need to be computed at any point.

4.2. Lagrange projection for imposing Dirichlet boundary functions

In the first test, we use element-level Lagrange projection to impose Dirichlet boundary conditions at \( y=0 \), where a function for both displacements is specified (see Fig. 6(a)). We find the element-level coefficients of the spline interpolant at the Dirichlet boundary by evaluating (42) in each boundary element. For assembling the vector of global boundary coefficients, we apply the weighted averaging scheme (38) and (39). Alternatively, we also run the same set of computations using global \( L^2 \) projection to impose Dirichlet boundary conditions. Figure 7 shows the convergence of the error in the \( L^2 \) norm

\[
\epsilon_{L^2} = \int_\Omega \| \mathbf{u}_{\text{num}} - \mathbf{u}_{\text{ex}} \| \, d\Omega \quad (48)
\]

for NURBS of polynomial degree 2 to 5 as the mesh is uniformly refined. In this context, the numerical solution \( \mathbf{u}_{\text{num}} \) denotes the displacement vector obtained from the finite element analysis and \( \mathbf{u}_{\text{ex}} \) are the exact displacements (44). We observe that the analysis results obtained with local Lagrange projection are indistinguishable from analysis results obtained with global \( L^2 \) projection. We conclude that local Lagrange projection preserves full accuracy and optimal rates of convergence when used for imposing Dirichlet boundary functions.

In addition, we compare the \( L^2 \) convergence of isogeometric analysis with the NURBS mesh to the convergence of standard FEA that uses the same mesh size, but discretizes the quarter annulus with nodal quadrilateral elements of polynomial degree \( p \). We observe that both IGA and FEA converge optimally, but in FEA the accuracy per degree of freedom is significantly lower than in IGA.
4.3. Lagrange projection for spline interpolation over the complete mesh

In the second test, we assume that we know the solution (44) and want to project it onto the NURBS mesh. In finite element analysis, this operation occurs for example when the mesh changes from one time step to the next (for example to adjust mesh adaptivity) and the solution represented by the basis functions of the old mesh needs to be projected on the basis functions of the new mesh. We compare the error in $L^2$ norm computed with global $L^2$ projection, where $u_{num}$ and $u_{ex}$ in (48) denote the known solution and the projected solution, respectively. Figure 8 illustrates the convergence of the $L^2$ error (48) for different polynomial degrees $p$ when the corresponding NURBS mesh is uniformly refined. We observe that the local Lagrange projection preserves the optimal convergence rates of the underlying
(a) Quadratic NURBS mesh.  
(b) Cubic NURBS mesh.  
(c) Quartic NURBS mesh.  
(d) Quintic NURBS mesh.

Figure 8: Convergence of global $L^2$ projection and local Lagrange projection with different averaging schemes for the projection of a known solution on NURBS meshes.

basis for all degrees and leads to results that are practically indistinguishable from global $L^2$ projection. We conclude that we can achieve the same accuracy with Lagrange projection as with global $L^2$ projection. For error levels close to machine accuracy the error curve of local projection seems to level off earlier than the error curve of global projection (see Fig. 8d). We think that this can be attributed to the conditioning of the Lagrange basis and the extraction operators. The same phenomenon was observed for Bézier projection [25].

5. Summary and conclusions

The main objective of spline extraction is to enable procedures for the element-level computation of local arrays (e.g., element stiffness matrices and load vectors) that can be
implemented in standard finite element codes. The established extraction technology to-date is Bézier extraction, which is based on the decomposition of smooth basis functions into $C^0$ Bernstein polynomials. In this paper, we introduced an alternative extraction technology that we call Lagrange extraction, which links a smooth spline basis to the standard nodal basis of $C^0$ Lagrange polynomials. Lagrange basis functions are interpolatory at element nodes. This opens the door for a straightforward construction of the Lagrange extraction operator by simple point-wise evaluation of spline basis function values at the element nodes. When we restrict ourselves to B-spline discretizations (no rational basis functions involved), the Lagrange extraction operator links the B-spline control points directly to Lagrange control points that are located on the smooth geometry. Lagrange control points can be handed over to standard finite element subroutines as nodal positions of corresponding Lagrange elements, so that nodal finite element subroutines can remain completely untouched. Lagrange extraction thus allows the implementation of isogeometric analysis in standard nodal finite element codes with very simple algorithms and minimal intrusion. This is particularly attractive for IGA beginners who are at a preliminary stage and want to run prototype computations to rapidly test the effect of smooth basis functions for their application.

Lagrange projection is an element-level local projection technique to represent a given function over the isogeometric mesh in terms of its smooth basis functions. Lagrange projection leverages collocation-type concepts to establish the inverse of the Lagrange extraction operator as a direct link between target function values at element nodes and element-level spline coefficients of a local spline interpolant. We adopted a weighted averaging scheme to assemble local spline coefficients into the vector of global coefficients of the global interpolant. Local Lagrange projection converges optimally and its accuracy is virtually indistinguishable from global $L^2$ projection.

From a more fundamental viewpoint, we feel that Lagrange extraction and projection can contribute to bridging the gap between isogeometric and standard FEA technologies that currently seems to exist. We think that it can lower potential hurdles for computational analysts to adopting isogeometric capabilities in their codes. It is our hope that Lagrange extraction can thus provide an additional pathway to the opportunities of higher order smoothness for a broad audience within the FEA community.

Acknowledgments. We acknowledge the Minnesota Supercomputing Institute (MSI) of the University of Minnesota for providing computing resources that have contributed to the research results reported within this paper (https://www.msi.umn.edu/).
Appendix A. Lagrange extraction: A gentle first approach to IGA

This Appendix is an add-on to Section 2.4. In the following, we provide some more details on how to use Lagrange extraction and projection to implement IGA capabilities with minimal intrusion in a standard nodal FEA code, opening the door for a gentle first approach to isogeometric analysis for IGA beginners. We restrict ourselves to single-patch B-spline discretizations in one, two and three dimensions. We first discuss very simple MATLAB functions, which can be used in a preprocessing step to generate element Lagrange extraction operators in 1D, 2D and 3D. We then describe how to generate approximate B-spline control points using the Lagrange projection operator, i.e. the inverse of the Lagrange extraction operator.

Appendix A.1. Generating element Lagrange extraction operators

Algorithm 1 generates element Lagrange extraction operators for a simple one-dimensional B-spline discretization of polynomial degree $p$ and number of elements $n$. It uses the following spline related MATLAB functions: (a) `augknot(:,p+1)` returns a nondecreasing knot sequence that has the first and last knot with exact multiplicity $p+1$ (open knot vector); (b) `spcol(knots,p+1,:)` evaluates the spline basis functions of polynomial degree $p$ and knot vector `knots` at the Lagrange nodes. Our code returns a three-dimensional array $D$, whose first two indices are the row and column index of the extraction operator and the third index denotes the element.

Algorithms 2 and 3 extend Algorithm 1 to two and three dimensions. Knot vectors and B-splines are evaluated for each parametric direction first and the Lagrange extraction operators are then constructed for each element by using the MATLAB tensor-product function `kron(:,:)`. We note that we number the tensor-product basis functions lexicographically, that is, we traverse the tensor-product structure first in $x$-direction, then in $y$-direction, and finally in $z$-direction (only in 3D). The element Lagrange extraction operators are initially

<table>
<thead>
<tr>
<th>Data:</th>
<th>Polynomial degree $p$, number of elements (i.e., knot spans) $n$;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result:</td>
<td>Lagrange extraction operator $D(:,;i_{ele})$ for each element $i_{ele}$ of the 1D spline patch;</td>
</tr>
</tbody>
</table>

D = LagrangeExtraction(p,n)

% Generate open knot vector
knots = augknt(0:1:n,p+1);

% Evaluate B-splines at equally spaced Lagrange nodes
N = transpose(spcol(knots,p+1,0:1/p:n));

% Construct Lagrange extraction operator from nodal B-spline evaluations
for i=1:n do
    D(:,;i) = N(i:i+p,p*(i-1)+1:p*i+1);
end

Algorithm 1:
Compute element Lagrange extraction operators for a 1D B-spline patch.
constructed for a scalar problem, where each basis function corresponds to one degree of freedom. For vector-valued problems (e.g., elasticity), each row of the initial extraction operator needs to be repeated times the number of degrees of freedom per basis function (and the row entries need to be shifted accordingly) to construct the corresponding extraction operator. In Algorithms 2 and 3, these operations are invoked by setting the Boolean variable \text{vec} to one, which generates Lagrange extraction operators for tensor-valued 2D and 3D elasticity problems, respectively.

Data: Polynomial degree \( p \), number of elements (i.e., knot spans) \( nx, ny \) in each parametric direction, Boolean \text{vec} (\text{vec} = 0: extraction operator for scalar problem; \text{vec} = 1: extraction operator for elasticity);

Result: Local Lagrange extraction operator \( D(:, :, i_{ele}) \) for each element \( i_{ele} \) of the 2D tensor-product spline patch;

\[
D = \text{LagrangeExtraction2D}(p, nx, ny, \text{vec})
\]

\[
\text{for } j=1:ny \text{ do}
\]
\[
\text{for } i=1:nx \text{ do}
\]
\[
D(:, (j-1)*nx + i) = \text{kron}(Ny(j:j+p, p*(j-1)+1:p*j+1), ... \text{Nx}(i:i+p, p*(i-1)+1:p*i+1));
\]
\[
\text{end}
\]
\[
\text{end}
\]

\[
\text{if } \text{vec} == 1 \text{ then}
\]
\[
\text{for } i_{ele} = 1:\text{size}(D,3) \text{ do}
\]
\[
\text{row} = 1;
\]
\[
\text{for } i=1:\text{size}(D,1) \text{ do}
\]
\[
Dvec(\text{row}, :, i_{ele}) = D(:, :, i_{ele}) * \text{kron}(\text{eye}((p+1)^2), [1 \ 0]);
\]
\[
Dvec(\text{row}+1, :, i_{ele}) = D(:, :, i_{ele}) * \text{kron}(\text{eye}((p+1)^2), [0 \ 1]);
\]
\[
\text{row} = \text{row}+2;
\]
\[
\text{end}
\]
\[
\text{D} = \text{Dvec};
\]
\[
\text{end}
\]

Algorithm 2:
Compute element Lagrange extraction operators for a 2D B-spline patch.
Appendix A.2. Generating element Lagrange control points without CAD

In addition to the element Lagrange extraction operators, we require corresponding spline control points that represent the geometry with the given spline basis. In the sense of the isogeometric paradigm, this information can be generated with a CAD tool and imported

Data: Polynomial degree \( p \), number of elements (i.e., knot spans) \( nx, ny, nz \) in each parametric direction, Boolean \( vec \) (\( vec = 0 \): extraction operator for scalar problem; \( vec = 1 \): extraction operator for elasticity);

Result: Local Lagrange extraction operator \( D(:, :, i_{ele}) \) for each element \( i_{ele} \) of the tensor-product spline patch;

\[
D = \text{LagrangeExtraction3D}(p, nx, ny, nz, vec)
\]

\[
\begin{align*}
\text{Generate open knot vector in each parametric direction} \\
\text{knots}_x &= \text{augknt}(0:1:nx,p+1); \\
\text{knots}_y &= \text{augknt}(0:1:ny,p+1); \\
\text{knots}_z &= \text{augknt}(0:1:nz,p+1);
\end{align*}
\]

\[
\begin{align*}
\text{Evaluate univariate B-splines at equally spaced Lagrange nodes in each parametric direction} \\
\text{Nx} &= \text{transpose}(\text{spcol}(\text{knots}_x,p+1,0:1/p:nx)); \\
\text{Ny} &= \text{transpose}(\text{spcol}(\text{knots}_y,p+1,0:1/p:ny)); \\
\text{Nz} &= \text{transpose}(\text{spcol}(\text{knots}_z,p+1,0:1/p:nz));
\end{align*}
\]

\[
\begin{align*}
\text{Construct Lagrange extraction operator with kron-function for each element} \\
\text{for } k=1:nz \text{ do} \\
\quad \text{for } j=1:ny \text{ do} \\
\quad \quad \text{for } i=1:nx \text{ do} \\
\quad \quad \quad \quad D(:, (k-1)*(ny*nx)+(j-1)*nx+i) = \text{kron}(\text{Nz}(k:k+p,p*(k-1)+1:p*k+1), ... \\
\quad \quad \quad \quad \text{kron}(\text{Ny}(j:j+p,p*(j-1)+1:p*j+1), ... \\
\quad \quad \quad \quad \text{Nx}(i:i+p,p*(i-1)+1:p*i+1))); \\
\quad \text{end} \\
\quad \text{end} \\
\text{end}
\end{align*}
\]

\[
\begin{align*}
\text{If required, extend to tensor-valued Lagrange extraction operator for 3D elasticity} \\
\text{if } vec==1 \text{ then} \\
\quad \text{for } i_{ele} = 1:\text{size}(D,3) \text{ do} \\
\quad \quad \text{row} = 1; \\
\quad \quad \text{for } i=1:\text{size}(D,1) \text{ do} \\
\quad \quad \quad \text{Dvec}(row, :, i_{ele}) = D(i,:, i_{ele})*\text{kron}(\text{eye}(p+1^3),[1 0 0]); \\
\quad \quad \quad \text{Dvec}(row+1,:, i_{ele}) = D(i,:, i_{ele})*\text{kron}(\text{eye}(p+1^3),[0 1 0]); \\
\quad \quad \quad \text{Dvec}(row+2,:, i_{ele}) = D(i,:, i_{ele})*\text{kron}(\text{eye}(p+1^3),[0 0 1]); \\
\quad \quad \quad \text{row} = \text{row}+3; \\
\quad \text{end} \\
\quad \text{end} \\
\text{D} = \text{Dvec}; \\
\text{end}
\end{align*}
\]

Algorithm 3:
Compute element Lagrange extraction operators for a 3D B-spline patch.
Standard elements and nodes.

B-spline elements and control points.

Figure A.9: A standard quadrilateral finite element mesh can be easily projected on a corresponding B-spline mesh that smoothly approximates a circular annulus.

Figure A.10: Geometry error of the approximate quadratic B-spline mesh in terms of absolute difference between the outer edge of the approximation and the exact outer radius \( R = 4 \). We plot this error along the outer edge and for three different meshes.

into the analysis code. However, beginners in IGA will most likely not have access to a working design-through-analysis pipeline that can provide this information. In this case, Lagrange projection offers a simple alternative to easily generate approximations of spline control points from the nodal coordinates of the element nodes. We note that if the geometry can be described by polynomial functions of degree \( p \) exactly, the geometry representations with spline control points and nodal coordinates are equivalent and both exact. Examples include lines in 1D, rectangles and parallelograms in 2D, and cubes and prisms in 3D.

In Fig. A.9, we illustrate the procedure for the quarter annulus. For representing the geometry exactly, we need a NURBS parameterization, from which we can generate a rational Lagrange parameterization if we want to use Lagrange extraction. In case there is no way for us to find NURBS control points and weights, we can generate a B-spline approximation
as follows. We first use increments of the radius $r$ and the angle $\theta$ given in Fig. 6a to find the nodal coordinates for each element, shown in Fig. A.9a for quadratic Lagrange elements. Plugging the nodal coordinates into (37), we find the corresponding local B-spline control points by local Lagrange projection and can subsequently assemble the global vector of B-spline control points with the averaging procedure (38) and (39). The global B-spline control points represent a smooth approximation of the exact circular geometry (see Fig. A.9b). Figure A.10 plots the geometry error in terms of the difference in radius from the exact value evaluated along the outer edge of the circular annulus. We observe that the geometry error is very small and rapidly decreases with mesh refinement, so that the B-spline mesh is a very good approximation of the exact geometry. Using Lagrange extraction (6) we can transfer the B-spline control points back in corresponding Lagrange control points that represent the smooth geometry approximation by $C^0$ nodal basis functions. We emphasize that the Lagrange control points are different from the initial nodal coordinates shown in Fig. A.9a, which represent a $C^0$ continuous geometry (i.e., this geometry has small kinks at element boundaries).
References


