Fast formation and assembly of finite element matrices with application to isogeometric linear elasticity

Article *in* Computer Methods in Applied Mechanics and Engineering · October 2019

DOI: 10.1016/j.cma.2019.06.020

CITATIONS
15

READS
218

5 authors, including:

Rene Hiemstra
Leibniz Universität Hannover
22 PUBLICATIONS **313** CITATIONS
SEE PROFILE

Mattia Tani
University of Pavia
24 PUBLICATIONS **265** CITATIONS
SEE PROFILE

Francesco Calabro
University of Naples Federico II
39 PUBLICATIONS **821** CITATIONS
SEE PROFILE

Thomas J.R. Hughes
University of Texas at Austin
538 PUBLICATIONS **77,171** CITATIONS
SEE PROFILE

Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 355 (2019) 234–260
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Fast formation and assembly of finite element matrices with application to isogeometric linear elasticity

René R. Hiemstra[a,*], Giancarlo Sangalli[b,c], Mattia Tani[c], Francesco Calabrò[d], Thomas J.R. Hughes[a]

[a] *Institute for Computational Engineering and Sciences (ICES), University of Texas at Austin, United States*
[b] *Dipartimento di Matematica "F. Casorati", Università di Pavia, Italy*
[c] *Istituto di Matematica Applicata e Tecnologie Informatiche "Enrico Magenes", Pavia, Italy*
[d] *Dipartimento di Matematica e Applicazioni "R. Caccioppoli", Università di Napoli "Federico II", Italy*

Received 18 March 2019; received in revised form 16 June 2019; accepted 17 June 2019
Available online xxxx

Highlights

- We extend the weighted quadrature scheme in [2] to accurately integrate the elements of the stiffness matrix in linear elasticity.
- We propose a means of distributing quadrature points for non-uniform, mixed continuity, spline space.
- We show how the stiffness matrix can be formed contiguously in the storage order of the sparse matrix, thereby minimizing memory overhead and eliminating the addition assignment operation on sparse matrices.
- We compare formation and assembly times of the proposed methodology with respect to standard finite element procedures and show that enormous savings are possible, especially at high polynomial order.

Abstract

Recently, a new formation and assembly strategy was proposed in [1], which resulted in significant speedups in the formation and assembly time of the Galerkin mass matrix in isogeometric analysis. The strategy relies on three key ingredients: (1) assembly row by row, instead of element by element; and an efficient formation strategy based on (2) sum factorization and (3) weighted quadrature, that is applied to each specific row of the matrix. Compared to traditional element procedures applied to three dimensional problems, the computational complexity is lowered from $\mathcal{O}(p^9)$ per degree of freedom to $\mathcal{O}(p^4)$, where p is the order of polynomials. This is close to the theoretical minimum of $\mathcal{O}(p^3)$, attained by, for example, collocation. Consequently, this type of formation and assembly scales favorably with polynomial degree, which opens the way for high order isogeometric analysis employing k-refinement, that is, use of maximally smooth, higher order splines. In this work we discuss various important details for the practical implementation of the weighted quadrature formation strategy proposed in [1]. Specifically, we extend the weighted quadrature scheme to accurately integrate the elements of the stiffness matrix in linear elasticity and propose a means of distributing quadrature points for non-uniform, mixed continuity, spline spaces. Furthermore, we discuss efficient access and assignment into the prevalent sparse matrix data structures, namely, Compressed Sparse Row (CSR) and Compressed Sparse Column (CSC). In particular, row-by-row or column-by-column assembly allows matrix rows or columns, respectively, to be formed contiguously in the storage order of the sparse matrix, thereby minimizing the memory

* Corresponding author.
E-mail addresses: rene@ices.utexas.edu (R.R. Hiemstra), giancarlo.sangalli@unipv.it (G. Sangalli), mattia.tani@imati.cnr.it (M. Tani), francesco.calabro@unina.it (F. Calabrò), hughes@ices.utexas.edu (T.J.R. Hughes).

https://doi.org/10.1016/j.cma.2019.06.020
0045-7825/

overhead and eliminating the addition assignment operation on sparse matrices. Several three-dimensional benchmark problems illustrate the efficiency and efficacy of the proposed formation and assembly technique applied to isogeometric linear elasticity. We show that the accuracy of full Gauss quadrature is maintained while the computational burden of forming the matrix equations is significantly reduced.

Keywords: Isogeometric analysis; Weighted Quadrature; Sum Factorization; Row assembly; Column assembly

1. Introduction

The great success of the finite element method [3,4] can be attributed to its strong theoretical rooting in the fields of variational calculus and functional analysis, and its wide applicability to a range of complex physical phenomena involving complex geometry. Arguably, however, the most important reason for success of the finite element method has been the rise of the computer age and the relative ease at which mathematical finite element formulations can be translated to efficient computer code.

At the core of any finite element implementation are the element subroutines that form the local element matrices that are subsequently assembled in the global system of matrix equations. Early on, it was recognized that this assembly process could be performed in an element-by-element fashion due to the local support of the finite element basis functions, which were of low order and low continuity. This approach worked very well in the early days of serial computation where floating point operations were expensive and memory limited. Today, floating point operations are relatively inexpensive, memory is abundant, but expensive to move, and serial computation has been replaced by parallel computation. Not only has the hardware changed. The finite element method has evolved too. It is therefore questionable whether the element-by-element assembly process is still "the right one".

The emerging field of isogeometric analysis [5,6] has cast new light on the finite element method. Although it was introduced to improve the interoperability across the design, analysis and manufacturing pipeline, isogeometric analysis has proven its fidelity as an analysis technology. In particular, several investigations have demonstrated superior robustness and accuracy per degree of freedom [6–8], and excellent spectral behavior has been shown in a number of studies [9–12] explaining the excellent results obtained in applications such as structural dynamics [5] and incompressible fluid dynamics [13,14]. Moreover, some fundamental properties of splines have led to the design of so called compatible discretization methods that preserve important mathematical structure of the partial differential equations under consideration [2,15,16].

Initially, isogeometric analysis was performed using standard finite element procedures. Especially important is the introduction of Bézier extraction [17,18], which allowed isogeometric analysis to be performed using minor alterations to an existing finite element code. This certainly allowed for the rapid dispersion of isogeometric analysis into the finite element community and has been important for its acceptance and application in existing finite element software such as LS-DYNA [19,20]. On the other hand, standard finite element formation and assembly practices currently limit isogeometric analysis to low polynomial order, primarily quadratics and cubics. This is unfortunate because a number of studies have shown that high order and high continuity combined in "k-refinement" benefit greatly the accuracy and, more surprisingly, the robustness of the method in a wide variety of applications in solid and fluid mechanics.

Many important contributions have been made recently in order to make high order and high continuity isogeometric analysis more efficient. Firstly, collocating the strong form of the equations [21–23] is a viable alternative to the Galerkin method that decreases the formation and assembly time by orders of magnitude at the expense of decreased accuracy per degree of freedom and sub-optimal convergence rates [24]. The recently introduced "variational collocation method" [25,26] puts the collocation method in a variational context and shows promise to render collocation as accurate as Galerkin. Several other works have focused on making Galerkin more efficient by employing specialized or reduced quadrature rules [27–36] and employing fast formation routines such as sum factorization [37–40] or table lookup [41], and employing a reordering of computations that benefits from GPU accelerated parallel computation [42].

The objective of this paper is to further develop the recently introduced formation and assembly strategy presented in [1,43]. Besides obvious advantages, such as parallel scalability, it turns out that this formation and assembly strategy scales favorably with polynomial degree p, which opens the way for high-order isogeometric analysis

Table 1
Number of floating point operations per degree of freedom for different formation and assembly strategies of the mass matrix in three-dimensional tensor product isogeometric analysis using smooth C^{p-1} splines. Here the c's are constants independent of p. Note that row assembly using a quadrature rule with $\mathcal{O}(1)$ points per element results in $\mathcal{O}(p^4)$ operations per degree of freedom in each of the three separate stages of the sum factorization process.

(a) $\mathcal{O}\left(p^3\right)$ quadrature points per element

		Formation	
		Quadrature loop	Sum factorization
Assembly	Element loop	$c \cdot p^9$	$c_1 \cdot p^5$ $+c_2 \cdot p^6$ $+c_3 \cdot p^7$
	Row/Column loop	$c \cdot p^9$	$c_1 \cdot p^7$ $+c_2 \cdot p^6$ $+c_3 \cdot p^5$

(b) $\mathcal{O}(1)$ quadrature points per element

		Formation	
		Quadrature loop	Sum factorization
Assembly	Element loop	$c \cdot p^6$	$c_1 \cdot p^2$ $+c_2 \cdot p^4$ $+c_3 \cdot p^6$
	Row/Column loop	$c \cdot p^6$	$c_1 \cdot p^4$ $+c_2 \cdot p^4$ $+c_3 \cdot p^4$

employing k-refinement. In this work we extend the weighted quadrature perspective presented in [1] to the stiffness matrix in linear elasticity and discuss important practical considerations in computing the numerical integrals, such as distribution of the quadrature points in case of mixed continuity, non-uniform spline spaces, efficient access and assignment into sparse matrix data structures, and extension to distributed and shared memory parallel computation.

1.1. Row or column formation and assembly by means of weighted quadrature and sum factorization

Each row of a Galerkin system matrix is associated with a locally supported test function, while each matrix column is associated with a trial function. Devising a quadrature rule that integrates products with a particular test function can lead to an efficient quadrature strategy for computing the entries of the corresponding matrix row. This, combined with sum factorization, is the essential idea of weighted quadrature as presented in [1]. We briefly recall the main ingredients and compare among formation and assembly techniques in Table 1.

1.1.1. Weighted quadrature

The obvious way to improve the performance of the formation and assembly process is to design specialized quadrature rules that require fewer evaluations. The increased smoothness of splines provides opportunities to do so while maintaining accuracy. For example, in [34] Generalized Gaussian rules have been designed, which require $\lceil \frac{2p-r}{2} \rceil$ points per element, where r is the regularity of the univariate degree p spline space. Additionally, one can employ reduced integration techniques such that the error due to quadrature is bounded by the discretization error, see [33]. Unfortunately, the number of evaluations of these techniques is proportional to p. Weighted quadrature is a new technique in which the asymptotic cost with k-refinement does not depend on p. A specialized quadrature rule is designed for each test function by incorporating the test function within the integral measure. It turns out that roughly $3^{n_{\text{sd}}}$ points are required per element to integrate the entries of the isogeometric stiffness matrix in linear elasticity, where n_{sd} is the number of space dimensions.

1.1.2. Sum factorization

Sum factorization was first introduced in [40] and is now considered the technique of choice to attain efficient formation of local element matrices in hp-finite elements [39,44–46]. It has been applied in the context of isogeometric analysis in [37,38]. In essence, sum factorization is a reordering of the computations in such a way to exploit the underlying tensor product of the test and trial spaces involved. This reordering transforms multidimensional integrals into a series of one-dimensional integrals, thereby reducing computational complexity in three dimensional tensor product isogeometric analysis from $\mathcal{O}\left(p^9\right)$ to $\mathcal{O}\left(p^7\right)$ per degree of freedom. Here we have assumed that full Gauss quadrature is used resulting in $(p+1)^3$ quadrature points per element.

1.1.3. Row or column loop

Combining sum factorization and weighted quadrature in a row or column loop significantly reduces the computational complexity of forming and assembling the matrix equations. Table 1 illustrated that all three ingredients, that is, (1) row-loop, (2) sum-factorization, and (3) weighted quadrature, are necessary to lower the computational complexity from $\mathcal{O}\left(p^9\right)$ to $\mathcal{O}\left(p^4\right)$. The critical observation is that row-formation using a quadrature rule with $\mathcal{O}(1)$ points per element results in the proper balancing of each of the three separate stages in the sum factorization process.

1.2. Outline

The outline of this paper is as follows. In Section 2 we develop weighted quadrature rules that are applicable to second order partial differential equations. In particular, we show how to distribute the quadrature points and compute the weights of weighted quadrature rules. In Section 3 we introduce the model problem and derive the matrix equations arising from discretization of linear elasticity. We then show how to form the entries of the stiffness matrix using sum factorization and weighted quadrature. In Section 4 we discuss implementational details with regard to sum factorization and sparse matrix access and assignment and discuss opportunities for shared and distributed parallel computation. The presented combination of techniques have been applied in a numerical implementation which leads to the results in Section 5. Finally, conclusions are drawn and recommendations for future work are made in Section 6.

2. Weighted quadrature

By incorporating the test-function within the integral measure it is possible to develop test function specific quadrature rules that, asymptotically with k-refinement, have $\mathcal{O}(1)$ points per degree of freedom, independent of the polynomial degree p. This results in formation and assembly procedures that scale nearly optimal with p. In this Section we introduce weighted quadrature in the general context of arbitrary second order partial differential equations. We start with a brief recapitulation of univariate splines and introduce the main properties that we use in the context of weighted quadrature.

2.1. Background and notation

A spline is a piecewise polynomial that is characterized by the polynomial degree of its segments and the prescribed regularity at their interfaces. A convenient basis in which to represent polynomial splines is given by B-splines [47,48]. Consider a partitioning of $\Delta = [\hat{x}_0, \hat{x}_\delta]$ into δ elements,

$$\hat{x}_0 < \hat{x}_1 < \cdots < \hat{x}_{k-1} < \hat{x}_k < \cdots < \hat{x}_\delta.$$

With every internal *breakpoint*, $\hat{x}_k$, we associate an integer, r_k, prescribing the regularity between the polynomial pieces. Then, given the *knot-multiplicity*, $\{p - r_k\}_{k=1}^{\delta-1}$, we can define the *knot vector* as,

$$\Xi = \{\xi_i\}_{i=1}^{d+p+1} := \{ \underbrace{\hat{x}_0, \ldots, \hat{x}_0}_{p+1}, \ldots, \underbrace{\hat{x}_{k-1}, \ldots, \hat{x}_{k-1}}_{p-r_{k-1}}, \underbrace{\hat{x}_k, \ldots, \hat{x}_k}_{p-r_k}, \ldots, \underbrace{\hat{x}_\delta, \ldots, \hat{x}_\delta}_{p+1} \}. \tag{1}$$

With a knot vector in hand, B-splines are stably and efficiently computed using the Cox–DeBoor recursion,

$$\hat{N}_{i,0}(\hat{x}) = \begin{cases} 1 & \text{if } \hat{x} \in [\xi_i, \xi_{i+1}) \\ 0 & \text{otherwise} \end{cases}$$

$$\hat{N}_{i,p}(\hat{x}) = \frac{\hat{x} - \xi_i}{\xi_{i+p} - \xi_i} \hat{N}_{i,p-1}(\hat{x}) + \frac{\xi_{i+p+1} - \hat{x}}{\xi_{i+p+1} - \xi_{i+1}} \hat{N}_{i+1,p-1}(\hat{x})$$

where, by definition, $0/0 = 0$.

Let $\boldsymbol{r} = \{0 \leq r_k \leq p-1,\ k = 1, \ldots, \delta - 1\}$. The space of polynomial splines of degree p and dimension d with r_k continuous derivatives at breakpoint $\hat{x}_k$ is defined as,

$$\mathbb{S}^p_{\boldsymbol{r}} = \text{span}\left(\hat{N}_{i,p}(\hat{x})\right)_{i=1}^{d}. \tag{2}$$

B-splines have important mathematical properties, many of which are useful in design as well as in analysis. B-spline basis functions of degree p may have up to $p-1$ continuous derivatives, they form a positive partition of unity, and have local support. Furthermore, they feature the following differentiation rule, which will prove important for our subsequent discussion,

$$\hat{N}'_{i,p}(\hat{x}) = \frac{p}{\xi_{i+p} - \xi_i} \hat{N}_{i,p-1}(\hat{x}) - \frac{p}{\xi_{i+p+1} - \xi_{i+1}} \hat{N}_{i+1,p-1}(\hat{x}). \tag{3}$$

2.2. Definition of weighted quadrature rules

The univariate integrals that we would like to efficiently evaluate are of the form,

$$\int_{\Delta} wu\ c(\hat{x}) d\hat{x}, \qquad \int_{\Delta} wu'\ c(\hat{x}) d\hat{x}, \tag{4}$$

$$\int_{\Delta} w'u\ c(\hat{x}) d\hat{x}, \qquad \int_{\Delta} w'u'\ c(\hat{x}) d\hat{x}. \tag{5}$$

Here $u, w \in \mathbb{S}^p_{\boldsymbol{r}}$ are trial and test functions and $c(\hat{x})$ is a coefficient that depends on the geometry mapping and the material behavior. These type of integrals and their multivariate generalizations occur in a wide variety of linear boundary value problems involving diffusive, advective and reactive behavior.

Quadrature rules are primarily designed to be exact in case $c(\hat{x}) = 1$. Such rules perform well, in general, when applied to a physical problem where $c(\hat{x})$ is sufficiently regular. The error arising due to quadrature can be bounded by the discretization error such that optimal rates of convergence can be guaranteed.[1] Weighted quadrature also proceeds in this way. The novelty, however, is that a quadrature rule is designed for each test function, or its derivative, separately. This is done by combining the test-function or its derivative within the integral measure. This gives rise to two types of quadrature rules: one is designed to approximate integrals in (4) and the other integrals of the form in (5).

Definition 2.1 (*Weighted Quadrature*)**.** Let $v \in \mathbb{S}^p_{\boldsymbol{r}-1}$ and $\hat{M}(\hat{x})$ a B-spline test function. We define weighted quadrature rules,

$$\mathcal{Q}^{(0)}_{\hat{M}}(v) = \sum_k v(\mathrm{x}_k) \cdot W^{(0)}_{\hat{M},k} := \int_{\Delta} v(\hat{x}) \left(\hat{M}(\hat{x}) d\hat{x}\right) \tag{6}$$

$$\mathcal{Q}^{(1)}_{\hat{M}}(v) = \sum_k v(\mathrm{x}_k) \cdot W^{(1)}_{\hat{M},k} := \int_{\Delta} v(\hat{x}) \left(\hat{M}'(\hat{x}) d\hat{x}\right) \tag{7}$$

where $\{\ \mathrm{x}_k \in [\hat{x}_0, \hat{x}_\delta],\ W^{(\alpha)}_{\hat{M},k} \in \mathbb{R},\ k = 1, 2, \ldots, q\ \}$.

Consider (4) and (5) with $w = \hat{M}$ and $c(\hat{x}) \equiv \text{const}$. Because both u, u' are elements of the space $\mathbb{S}^p_{\boldsymbol{r}-1}$, the quadrature rule $\mathcal{Q}^{(0)}_{\hat{M}}(\cdot)$ is exact for integrals of the form in (4), while $\mathcal{Q}^{(1)}_{\hat{M}}(\cdot)$ is exact for integrals of the form in (5).

[1] Optimal rates of convergence can also be attained under looser restrictions, e.g. when strategies such as reduced integration are applied, see [33].

2.2.1. Computation of weighted quadrature rules $\mathcal{Q}^{(0)}_{\hat{M}}(\cdot)$

Let $\{\ \hat{N}_{j,p}(\hat{x})\ \}_{j=1}^{d}$ denote a B-spline basis for the target space $\mathbb{S}^p_{\boldsymbol{r}-1}$ and $\hat{M}(\hat{x}) \in \mathbb{S}^p_{\boldsymbol{r}}$ a B-spline test function. Note that $\mathbb{S}^p_{\boldsymbol{r}} \subset \mathbb{S}^p_{\boldsymbol{r}-1}$ and $\mathbb{S}^{p-1}_{\boldsymbol{r}-1} \subset \mathbb{S}^p_{\boldsymbol{r}-1}$, that is, the trial space and the space spanned by derivatives of the trial functions are contained within the target space. From (6)–(7) we can derive the following equations for exact quadrature,

$$\begin{aligned}
\mathcal{Q}^{(0)}_{\hat{M}}\left(\hat{N}_{j_1}\right) &= \sum_{k=k_1}^{k_2} W^{(0)}_{\hat{M},k} \cdot \hat{N}_{j_1}(\mathrm{x}_k) := \int_{\Delta} \hat{N}_{j_1}(\hat{x}) \left(\hat{M}(\hat{x}) d\hat{x}\right) \\
&\vdots \qquad\qquad\qquad\qquad \vdots \\
\mathcal{Q}^{(0)}_{\hat{M}}\left(\hat{N}_{j_2}\right) &= \sum_{k=k_1}^{k_2} W^{(0)}_{\hat{M},k} \cdot \hat{N}_{j_2}(\mathrm{x}_k) := \int_{\Delta} \hat{N}_{j_2}(\hat{x}) \left(\hat{M}(\hat{x}) d\hat{x}\right).
\end{aligned} \tag{8}$$

Here $j_1, \ldots, j_2$ are the indices of trial functions that have non-zero intersecting support with test function $\hat{M}$ and $k_1, \ldots, k_2$ refer to the quadrature points that lie within the support of test function $\hat{M}$. Quadrature weights corresponding to $k \notin k_1, \ldots, k_2$ are set to zero.

If the quadrature points $\{\mathrm{x}_k,\ k = k_1, \ldots, k_2\}$ are chosen a priori in such a way that the problem statement is well defined, that is, the quadrature rule exists, then the above exactness conditions are linear in the weights and are simple to solve. The solution may be non-unique, however, because the number of quadrature weights (unknowns) is generally greater than the number of exactness conditions. Therefore, we compute the non-zero weights of the quadrature rule,

$$\mathbf{w} = \{\ W^{(0)}_{\hat{M},k},\ k = k_1, \ldots, k_2\ \},$$

by means of the quadratic optimization problem in (9). This is a weighted least squares problem where the matrix Z serves to normalize the unknown weights in $\mathbf{w}$, which aids positivity and boundedness. The solution of the stated problem is linear in the weights and can be stably and efficiently computed using QR-factorization.

(O)

Minimize,

$$P(\mathbf{w}) = \frac{1}{2}\|\mathsf{Z}^{-1}\mathbf{w}\|^2_{l^2} \tag{9a}$$

over all $\mathbf{w} \in \mathbb{R}^{q_o}$ subject to,

$$\mathsf{A}\mathbf{w} = \mathbf{b} \tag{9b}$$

where $\mathsf{A} \in \mathbb{R}^{d_o \times q_o}$ and $\mathbf{b} \in \mathbb{R}^{d_o}$ are defined as,

$$\mathsf{A} = \begin{pmatrix} \hat{N}_{j_1}(\mathrm{x}_{k_1}) & \cdots & \hat{N}_{j_1}(\mathrm{x}_{k_2}) \\ \vdots & & \vdots \\ \hat{N}_{j_2}(\mathrm{x}_{k_1}) & \cdots & \hat{N}_{j_2}(\mathrm{x}_{k_2}) \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} \int \hat{N}_{j_1}(\hat{x}) \left(\hat{M}(\hat{x}) d\hat{x}\right) \\ \vdots \\ \int \hat{N}_{j_2}(\hat{x}) \left(\hat{M}(\hat{x}) d\hat{x}\right) \end{pmatrix} \tag{9c}$$

and,

$$\mathsf{Z} = \mathrm{diag}\left(\hat{M}(\mathrm{x}_{k_1})h_{k_1}, \ldots, \hat{M}(\mathrm{x}_{k_2})h_{k_2}\right), \qquad h_k = \mathrm{x}_{k+1/2} - \mathrm{x}_{k-1/2}. \tag{9d}$$

Remark 2.2. To maintain 16 digits of accuracy in the quadrature rule the weights have to be computed in high precision. We solve the normal equations using QR factorization with arbitrary precision arithmetic provided by the Julia language.

Remark 2.3. The univariate rules are precomputed, stored and subsequently used in a tensor product isogeometric discretization. The cost of solving for the weights is at most $\mathcal{O}(p^3)$ and is negligible compared to the formation and assembly costs.

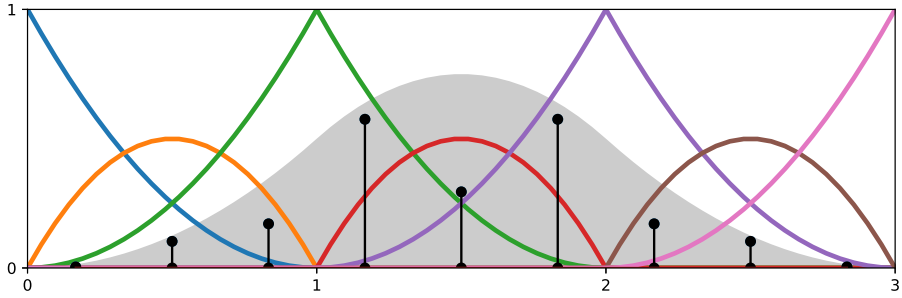


Fig. 1. A weighted quadrature rule corresponding to a uniform quadratic B-spline test function, denoted in grey, from Example 2.5. The remaining functions form a basis for the target space $\mathbb{S}^2_{\boldsymbol{r}-1}$. The quadrature points are chosen a priori and the corresponding weights, that is, the vertical coordinates, are computed by solving the linear equations in (9).

Remark 2.4. Instead of the presented least squares approach one could solve a linear programming problem, using e.g. the simplex method, to enforce positivity [49]. However, this fails if such a rule does not exist given the chosen distribution of quadrature points.

Example 2.5 (*Weighted Quadrature*)**.** Let $\Delta = \{\, 0, 1, 2, 3 \,\}$ with $\boldsymbol{r} = \{\, 1, 1 \,\}$. Consider a target space for weighted quadrature, $\mathbb{S}^2_{\boldsymbol{r}-1} = \operatorname{span}\left(\hat{N}_{j,2}\right)_{j=1}^{7}$, and a test space, $\mathbb{S}^2_{\boldsymbol{r}} = \operatorname{span}\left(\hat{M}_{i,2}\right)_{i=1}^{5}$. We consider the problem of determining a weighted quadrature rule $\mathcal{Q}^{(0)}_{\hat{M}_{3,2}}(\cdot)$, associated with $\hat{M}_{3,2}(\hat{x})$, depicted in greyin Fig. 1.

The quadrature points have been chosen a priori as $\{\, \frac{1}{6}, \frac{1}{2}, \frac{5}{6}, 1+\frac{1}{6}, 1+\frac{1}{2}, 1+\frac{5}{6}, 2+\frac{1}{6}, 2+\frac{1}{2}, 2+\frac{5}{6} \,\}$. This results in the matrices A, Z and right-hand-side vector $\mathbf{b}$,

$$\mathsf{A} = \tfrac{1}{36}\begin{bmatrix} 25 & 9 & 1 & & & & & & \\ 10 & 18 & 10 & & & & & & \\ 1 & 9 & 25 & 25 & 9 & 1 & & & \\ & & & 10 & 18 & 10 & & & \\ & & & 1 & 9 & 25 & 25 & 9 & 1 \\ & & & & & & 10 & 18 & 10 \\ & & & & & & 1 & 9 & 25 \end{bmatrix}, \qquad \mathsf{Z} = \tfrac{1}{3\cdot 72}\operatorname{diag}\begin{bmatrix} 1 \\ 9 \\ 25 \\ 46 \\ 54 \\ 46 \\ 25 \\ 9 \\ 1 \end{bmatrix}, \qquad \mathbf{b} = \tfrac{1}{60}\begin{bmatrix} 1 \\ 3 \\ 19 \\ 14 \\ 19 \\ 3 \\ 1 \end{bmatrix}.$$

The solution, computed by solving (9), is depicted in Fig. 1 and in the first column of Table 2.

2.2.2. *Computation of weighted quadrature rules* $\mathcal{Q}^{(1)}_{\hat{M}}(\cdot)$

Using (3) and the fact that quadrature is a linear operation we obtain that,

$$\begin{aligned} \mathcal{Q}^{(1)}_{\hat{M}_{i,p}}(v) &= \int_{\Delta} v(\hat{x}) \left(\hat{M}'_{i,p}(\hat{x}) d\hat{x}\right) \\ &= \frac{p}{\xi_{i+p} - \xi_i} \int_{\Delta} v(\hat{x}) \left(\hat{M}_{i,p-1}(\hat{x}) d\hat{x}\right) - \frac{p}{\xi_{i+p+1} - \xi_{i+1}} \int_{\Delta} v(\hat{x}) \left(\hat{M}_{i+1,p-1}(\hat{x}) d\hat{x}\right) \\ &= \frac{p}{\xi_{i+p} - \xi_i} \mathcal{Q}^{(0)}_{\hat{M}_{i,p-1}}(v) - \frac{p}{\xi_{i+p+1} - \xi_{i+1}} \mathcal{Q}^{(0)}_{\hat{M}_{i+1,p-1}}(v). \end{aligned} \tag{10}$$

Consequently, we can simply compute weighted quadrature rules, $\{\, \mathcal{Q}^{(0)}_{\hat{M}_{i,p-1}}(\cdot),\ i = 1, 2, \ldots, d-1 \,\}$, with the technique presented in the previous section, and the rules $\{\, \mathcal{Q}^{(1)}_{\hat{M}_{i,p}}(\cdot),\ i = 1, 2, \ldots, d \,\}$ follow as linear combinations.

Table 2
Quadrature rules corresponding to Examples 2.5 and 2.6 computed up to 15 digits of accuracy.

k	x_k	$\mathcal{Q}^{(0)}_{\hat{M}_{3,2}}(\cdot)$	$\mathcal{Q}^{(0)}_{\hat{M}_{3,1}}(\cdot)$	$\mathcal{Q}^{(0)}_{\hat{M}_{4,1}}(\cdot)$	$\mathcal{Q}^{(1)}_{\hat{M}_{3,2}}(\cdot)$
1	$\frac{1}{6}$	0.002079195717828	0.062500000000000		0.062500000000000
2	$\frac{1}{2}$	0.051402680940575	0.125000000000000		0.125000000000000
3	$\frac{5}{6}$	0.085395978589138	0.312500000000000		0.312500000000000
4	$1\frac{1}{6}$	0.287524825693034	0.312500000000000	0.062500000000000	0.250000000000000
5	$1\frac{1}{2}$	0.147194638118850	0.125000000000000	0.125000000000000	0.000000000000000
6	$1\frac{5}{6}$	0.287524825693035	0.062500000000000	0.312500000000000	−0.250000000000000
7	$2\frac{1}{6}$	0.085395978589138		0.312500000000000	−0.312500000000000
8	$2\frac{1}{2}$	0.051402680940574		0.125000000000000	−0.125000000000000
9	$2\frac{5}{6}$	0.002079195717828		0.062500000000000	−0.062500000000000

Example 2.6 (*Weighted Quadrature*)**.** We continue the previous example. We consider the problem of determining the weighted quadrature rule $\mathcal{Q}^{(1)}_{\hat{M}_{3,2}}(\cdot)$. According to (10) this quadrature rule is computed as,

$$\mathcal{Q}^{(1)}_{\hat{M}_{3,2}}(\cdot) = \mathcal{Q}^{(0)}_{\hat{M}_{3,1}}(\cdot) - \mathcal{Q}^{(0)}_{\hat{M}_{4,1}}(\cdot)\,. \tag{11}$$

Due to the uniformity of the spline spaces and layout of the quadrature points the rules $\mathcal{Q}^{(0)}_{\hat{M}_{3,1}}(\cdot)$ and $\mathcal{Q}^{(0)}_{\hat{M}_{4,1}}(\cdot)$ are identical. Their solution, depicted in Table 2 and Fig. 2, follows from (9) where the matrices have been computed as,

$$\mathsf{A} = \tfrac{1}{36}\begin{bmatrix} 25 & 9 & 1 & & & \\ 10 & 18 & 10 & & & \\ 1 & 9 & 25 & 25 & 9 & 1 \\ & & & 10 & 18 & 10 \\ & & & 1 & 9 & 25 \end{bmatrix}, \qquad \mathsf{Z} = \tfrac{1}{3\cdot 6}\begin{bmatrix} 1 & & & & & \\ & 3 & & & & \\ & & 5 & & & \\ & & & 5 & & \\ & & & & 3 & \\ & & & & & 1 \end{bmatrix}, \qquad \mathbf{b} = \tfrac{1}{12}\begin{bmatrix} 1 \\ 2 \\ 6 \\ 2 \\ 1 \end{bmatrix}.$$

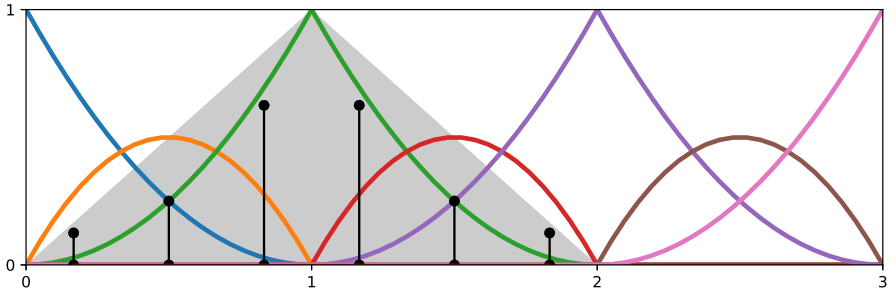
2.3. Layout and distribution of the quadrature points

As in [1] we consider weighted quadrature rules where the points are fixed a priori. This needs to be done in such a way that the problem statement in (9) is well defined, that is, matrix $\mathsf{A} \in \mathbb{R}^{d_o \times q_o}$ is of full rank and $q_o \geq d_o$. This poses constraints on the position of the quadrature points, as a direct consequence of the Schoenberg–Whitney theorem [47].

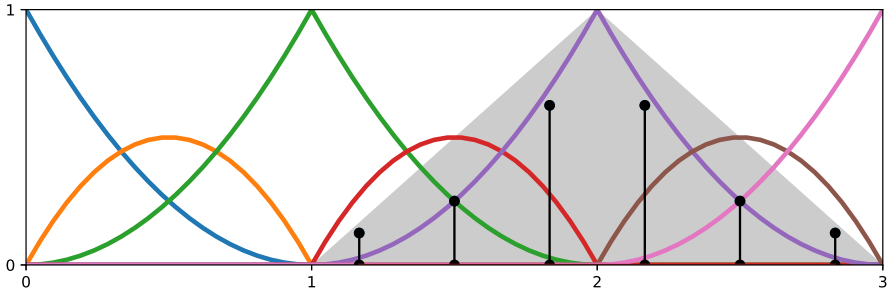
Remark 2.7. The Schoenberg–Whitney condition is fulfilled when every B-spline target function, $\hat{N}_j$, $j = j_1, \ldots, j_2$, can be uniquely associated with a quadrature point that lies within its support. In this work we additionally require that quadrature points $\{\, \mathrm{x}_k,\ k = k_1, \ldots, k_2 \,\}$ lie within the support of test function $\hat{M}$, although that is not strictly necessary for the invertibility of A.

In the following we introduce a procedure to determine the minimal number of quadrature points required in each element $I_k = [x_{k-1}, x_k)$, $k = 1, \ldots, \delta$. That the approach satisfies the Schoenberg–Whitney condition with the minimal number of quadrature points is left for future work. The process is delineated in Fig. 3 and uses the same target and test space as in Example 2.5. To refer to unions of several elements $I_r \cup \cdots \cup I_s$ we use the shorter notation $(r - s)$.

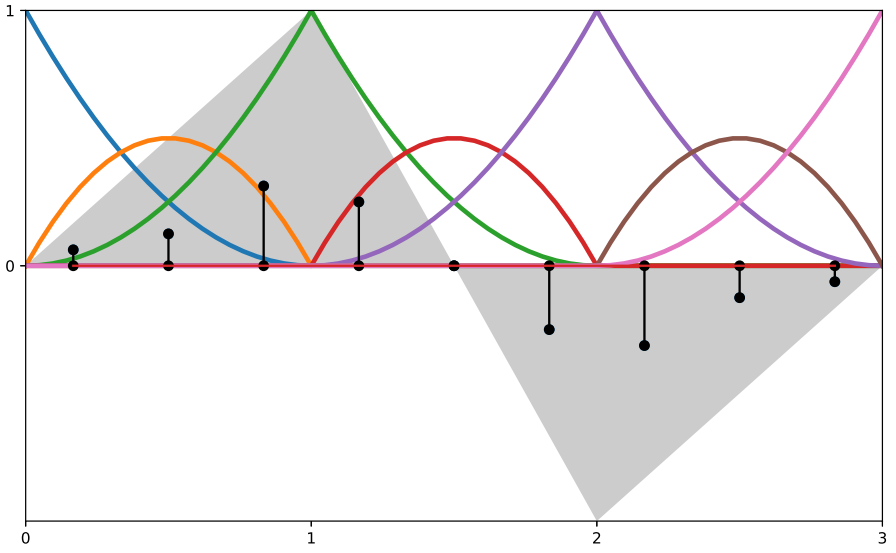
(**S1**) The first step is to determine the elements that are contained within the intersection of test and target functions, $\operatorname{supp}\left(\hat{N}_j\right) \cap \operatorname{supp}\left(\hat{M}_i\right)$. For example, the intersections of the target functions and test function depicted in



(a) Quadrature rule $\mathcal{Q}^{(0)}_{\hat{M}_{3,1}}(\cdot)$



(b) Quadrature rule $\mathcal{Q}^{(0)}_{\hat{M}_{4,1}}(\cdot)$



(c) Quadrature rule $\mathcal{Q}^{(1)}_{\hat{M}_{3,2}}(\cdot) = \mathcal{Q}^{(0)}_{\hat{M}_{3,1}}(\cdot) - \mathcal{Q}^{(0)}_{\hat{M}_{4,1}}(\cdot)$

Fig. 2. Weighted quadrature rules corresponding to derivatives of test functions of degree p can be written as a linear combination of weighted rules corresponding to test functions of degree $p-1$.

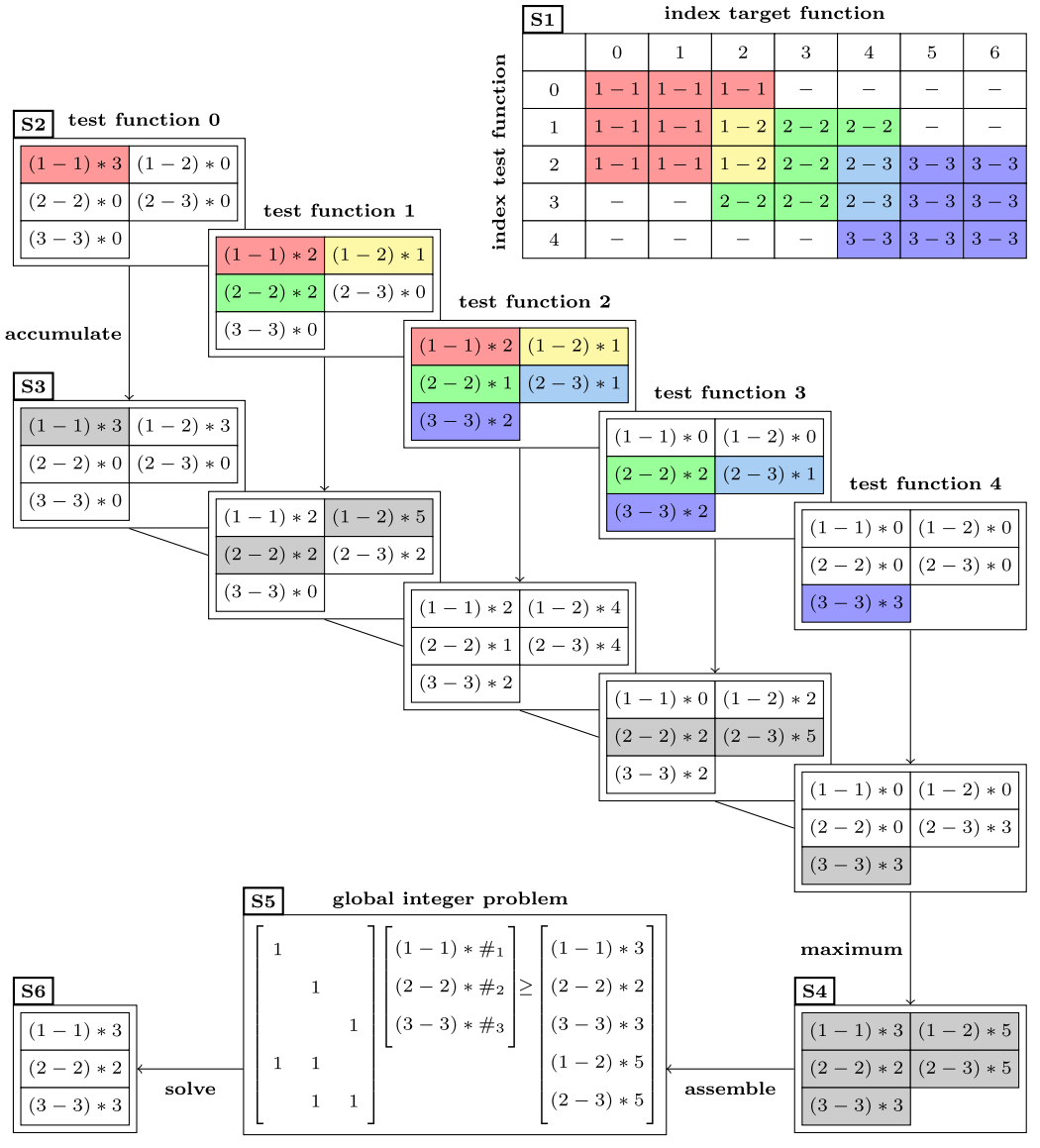


Fig. 3. Schematic illustrating the selection of a suitable layout of the quadrature points corresponding to the same test space $\mathbb{S}^p_{\boldsymbol{r}}$ and target space $\mathbb{S}^p_{\boldsymbol{r}-1}$ as used in Example 2.5, defined on $\Delta = \{ 0, 1, 2, 3 \}$, $p = 2$ and $\boldsymbol{r} = \{ 1, 1 \}$. First, the supports (in terms of elements) are determined for all the intersections between the test functions and the target functions (**S1**). The notation $(r - s)$ is short for $I_r \cup \cdots \cup I_s$. Note that common intersecting intervals are highlighted in the same color to display their equivalence. This information is stored in a three-dimensional array (**S2**), where each slice shown corresponds to a single test function. Subsequently, in every slice, the cumulative number of points in every interval is determined (**S3**). In the second slice, for example, the cumulative number of points in elements $1 - 2$ is 5; 2 points are located in element 1 (red), 2 points in element 2 (green) and 1 additional point in element 1 or 2 (yellow). By taking the maximum over the slices, we obtain the minimum number of points required in the union of $k = 1, 2$ elements (**S4**). These global constraints can be assembled into a global linear system of equations, (**S5**), with integer unknowns. Its global solution in (**S6**) provides the layout of the quadrature points.. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 1 are,

$$\operatorname{supp}\left(\hat{N}_1\right) \cap \operatorname{supp}\left(\hat{M}_3\right) \equiv \operatorname{supp}\left(\hat{N}_2\right) \cap \operatorname{supp}\left(\hat{M}_3\right) = I_1 = (1-1)$$
$$\operatorname{supp}\left(\hat{N}_3\right) \cap \operatorname{supp}\left(\hat{M}_3\right) = I_1 \cup I_2 = (1-2)$$

$$\text{supp}\left(\hat{N}_4\right) \cap \text{supp}\left(\hat{M}_3\right) = I_2 = (2-2)$$
$$\text{supp}\left(\hat{N}_5\right) \cap \text{supp}\left(\hat{M}_3\right) = I_2 \cup I_3 = (2-3)$$
$$\text{supp}\left(\hat{N}_6\right) \cap \text{supp}\left(\hat{M}_3\right) \equiv \text{supp}\left(\hat{N}_7\right) \cap \text{supp}\left(\hat{M}_3\right) = I_3 = (3-3).$$

This information is stored as row two in Fig. 3. Note that common intersecting intervals are highlighted in the same color to display their equivalence.

(**S2**) The information gathered in **S1** is reordered into a three-dimensional array, see Fig. 3. Each slice is associated with a test function. The notation $(r-s)*\mu_{rs}$ means that the interval $(r-s)$ has multiplicity μ_{rs} in a particular row of **S1**.

(**S3**) For each slice the minimum number of points in interval $(r-s)$, denoted by $\overline{\mu}_{rs}$, is determined by summing the multiplicities corresponding to intervals $(k-l) \subset (r-s)$, that is,

$$\overline{\mu}_{rs} = \sum_{k=r}^{s} \sum_{l=k}^{s} \mu_{kl}.$$

(**S4**) The global number of points required in each of the intervals is obtained by taking the maximum over each of the constraints in **S3**, highlighted in grey.

(**S5**) The constraints in **S4** lead to a global integer optimization problem. The unknowns are the number of quadrature points in each element.

(**S6**) Solve the integer problem and obtain layout.

In practice we solve the global integer problem using a greedy approach. Note that the first column in (**S4**), Fig. 3, provides a good initial guess. We incrementally increase the point count, distributed as evenly as possible, until all constraints in (**S5**) are satisfied.

Once the layout, that is, the number of points per element, is determined, points can be distributed in different ways. In this work we place points within element interiors only, and we uniformly distribute them as follows. Let $\hat{q}_i$ denote the number of points, $\mathrm{x}_{i1}, \ldots, \mathrm{x}_{i\hat{q}_i}$, to be distributed in element $(\hat{x}_{i-1}, \hat{x}_i)$. We define,

$$\mathrm{x}_{ik} = \left(1 - \frac{k}{2\hat{q}_i}\right) \hat{x}_{i-1} + \frac{k}{2\hat{q}_i} \hat{x}_i, \qquad k = 1, 3, 5 \ldots, 2\hat{q}_i - 1. \tag{12}$$

Performing this operation for every element in the partition, Δ, we obtain a set of quadrature points that yields well-defined weighted quadrature rules for the target space and test space under consideration.

3. Row formation by means of sum factorization and weighted quadrature

In this section we apply row formation and assembly by means of weighted quadrature and sum factorization to the equations of linear elasticity. Sum factorization constitutes a reordering of the computations that reduces the computational cost by a factor of p^2. Combined with row formation and assembly using weighted quadrature, this is further reduced to $\mathcal{O}(p^4)$ floating point operations per degree of freedom, which is close to the optimal, $\mathcal{O}(p^3)$, obtained by collocation.

3.1. The model problem

The ideas in this paper will be illustrated by considering linear elasticity as a model problem. Our notation and exposition closely follows that of [3]. We adopt the summation convention using indices $\mathrm{i}, \mathrm{j}, \mathrm{k}, \mathrm{l}$ that take values $1, 2, \ldots, n_{\mathrm{sd}}$, where $n_{\mathrm{sd}}(= 2 \text{ or } 3)$ denotes the number of space dimensions. Please note that these are Roman fonts and not to be confused with the earlier usage of Latin fonts. The partial derivative of a field w_{i} in direction j shall be denoted by $w_{\mathrm{i,j}}$. Multivariate tensor product B-spline basis functions will be denoted by,

$$\hat{\mathrm{N}}_A\left(\hat{x}\right) := \prod_{\mathrm{k}=1}^{n_{\mathrm{sd}}} \hat{N}_{a_{\mathrm{k}}}\left(\hat{x}_{\mathrm{k}}\right), \tag{13}$$

where node A corresponds to the product of one-dimensional splines indexed by $a_1, \ldots, a_{n_{\text{sd}}}$. Although we start with a standard presentation of the material, special attention is devoted to obtain the matrix equations in a new format that is suitable for applying sum factorization.

3.1.1. Classical elastostatics

Let $\Omega \subset \mathbb{R}^{n_{\text{sd}}}$ be an open set with piecewise smooth boundary $\Gamma = \overline{\Gamma_{g_i} \cup \Gamma_{h_i}}$, $\emptyset = \Gamma_{g_i} \cap \Gamma_{h_i}$, and outward unit normal vector n_i. The classical linear equations of elastostatics are,

$$\sigma_{ij,j} + f_i = 0 \quad \text{in } \Omega \tag{14a}$$

$$u_i = g_i \quad \text{on } \Gamma_{g_i} \tag{14b}$$

$$\sigma_{ij} \cdot n_j = h_i \quad \text{on } \Gamma_{h_i} \tag{14c}$$

Here σ_{ij} denote the Cartesian components of the Cauchy stress tensor, u_i are components of the displacement vector and f_i is the i'th component of the prescribed body force per unit of volume. The functions g_i and h_i are the prescribed boundary displacements and tractions, respectively.

The stress is related to the deformation through the generalized Hooke's law,

$$\sigma_{ij} = c_{ijkl} u_{(k,l)} \tag{15}$$

where c_{ijkl} are the components of the 4th order elasticity tensor, which is assumed symmetric positive definite on symmetric tensors, and $u_{(k,l)}$ denotes the symmetric part of the gradient of displacement, that is, $u_{(k,l)} = \frac{1}{2}\left(u_{k,l} + u_{l,k}\right)$. We shall assume the standard material law for a linear isotropic material, that is, $c_{ijkl}(x) = \mu(x)\left(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}\right) + \lambda(x)\delta_{ij}\delta_{kl}$, where $\lambda(x)$ and $\mu(x)$ are the Lamé parameters.

Restricting our attention to the $n_{\text{sd}} = 3$ case, we may write the constitutive law in (15) using Voigt notation as,

$$\boldsymbol{\sigma} = \mathsf{C}\, \boldsymbol{\epsilon}(\mathbf{u}) \tag{16}$$

where,

$$\mathsf{C} = \begin{bmatrix} c_{1111} & c_{1122} & c_{1133} & 0 & 0 & 0 \\ & c_{2222} & c_{2233} & 0 & 0 & 0 \\ & & c_{3333} & 0 & 0 & 0 \\ & & & c_{2323} & 0 & 0 \\ & & & & c_{1313} & 0 \\ \text{sym.} & & & & & c_{1212} \end{bmatrix}, \qquad \sigma = \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{23} \\ \sigma_{13} \\ \sigma_{12} \end{bmatrix}, \qquad \epsilon\left(\mathbf{u}\right) = \begin{bmatrix} u_{1,1} \\ u_{2,2} \\ u_{3,3} \\ u_{2,3} + u_{3,2} \\ u_{1,3} + u_{3,1} \\ u_{1,2} + u_{2,1} \end{bmatrix}. \tag{17}$$

3.1.2. The weak formulation and Galerkin's method

We proceed as is standard in finite elements and seek the displacement in a trial solution space $\mathcal{S} = \{\mathbf{u} \mid u_i \in \mathcal{S}_i\}$, with components, $\mathcal{S}_i := \left\{u_i \in H^1(\Omega) \mid u_i = g_i \text{ on } \Gamma_{g_i}\right\}$. Furthermore, let $\mathcal{V}_i := H^1_0(\Omega)$ be the space of functions in $H^1(\Omega)$ with homogeneous Dirichlet boundary conditions on Γ_{g_i} and define $\mathcal{V} = \{\mathbf{w} \mid w_i \in \mathcal{V}_i\}$. With these definitions the corresponding weak form of the problem in (14) is given as follows.

(W)

Find $\mathbf{u} \in \mathcal{S}$, such that,

$$a(\mathbf{w}, \mathbf{u}) = (\mathbf{w}, \mathbf{f}) + (\mathbf{w}, \mathbf{h})_\Gamma \tag{18a}$$

for all $\mathbf{w} \in \mathcal{V}$, where,

$$a(\mathbf{w}, \mathbf{u}) = \int_\Omega \epsilon(\mathbf{w})^T \mathsf{C}\, \epsilon(\mathbf{u}) d\Omega \tag{18b}$$

$$(\mathbf{w}, \mathbf{f}) = \int_\Omega w_i f_i d\Omega \tag{18c}$$

$$(\mathbf{w}, \mathbf{h})_\Gamma = \sum_{i=1}^{n_{\text{sd}}} \left(\int_{\Gamma_{h_i}} w_i h_i d\Gamma \right). \tag{18d}$$

Let $\mathcal{S}^h$ and $\mathcal{V}^h$ denote finite dimensional subspaces of $\mathcal{S}$ and $\mathcal{V}$, respectively. The displacement $\mathbf{u}^h \in \mathcal{S}^h$ can be decomposed, $\mathbf{u}^h = \mathbf{v}^h + \mathbf{g}^h$, such that $\mathbf{v}^h \in \mathcal{V}^h$ and $\mathbf{g}^h \in \mathcal{S}^h$ satisfies the Dirichlet boundary data. The Galerkin formulation reads,

$$(G) \quad \left\{ \begin{array}{l} \text{Find } \mathbf{v}^h \in \mathcal{S}^h \text{ such that for all } \mathbf{w}^h \in \mathcal{V}^h, \\ \qquad a(\mathbf{w}^h, \mathbf{v}^h) = \big(\mathbf{w}^h, \mathbf{f}\big) + \big(\mathbf{w}^h, \mathbf{h}\big)_{\Gamma} - a(\mathbf{w}^h, \mathbf{g}^h). \end{array} \right. \tag{19}$$

This presentation of the strong (S), weak (W) and Galerkin (G) form is standard, see [3]. In the following we derive the matrix equations in a form that is suitable for use with sum factorization.

3.1.3. The matrix formulation

Let $F : \hat{\Omega} \mapsto \Omega$ denote an n_{sd}-variate conforming parameterization of the physical domain $\Omega \subset \mathbb{R}^{n_{\text{sd}}}$ that maps each point $\hat{x} = (\hat{x}_1, \ldots, \hat{x}_{n_{\text{sd}}})$ in $\hat{\Omega}$ to a point $x = (x_1, \ldots, x_{n_{\text{sd}}})$ in physical space Ω. We assume that F is a smooth mapping with a smooth inverse, such that the Jacobian matrix $\big[DF(\hat{x})\big]_{\text{ij}} := \frac{\partial F_{\text{i}}}{\partial \hat{x}^{\text{j}}}$ and its inverse are well defined throughout Ω.

Let $\mathcal{S}_{\text{i}} \ni u^h_{\text{i}}(x) = v^h_{\text{i}}(x) + g^h_{\text{i}}(x)$ be expanded in terms of basis functions, $\mathrm{N}_A(x) = \hat{\mathrm{N}}_A \circ F^{-1}$, as,

$$v^h_{\text{i}}(x) = \sum_{A \in \eta - \eta_{g_{\text{i}}}} \mathrm{N}_A(x) \mathrm{v}_{A\text{i}} \tag{20}$$

$$g^h_{\text{i}}(x) = \sum_{A \in \eta_{g_{\text{i}}}} \mathrm{N}_A(x) \mathrm{g}_{A\text{i}}. \tag{21}$$

Similarly, an element of $\mathcal{V}^h_{\text{i}}$ has the form,

$$w^h_{\text{i}}(x) = \sum_{A \in \eta - \eta_{g_{\text{i}}}} \mathrm{N}_A(x) \mathrm{w}_{A\text{i}}. \tag{22}$$

Here $\eta = \{ 1, 2, \ldots, n_{\text{nodes}} \}$ denotes the set of global node numbers and $n_{\text{nodes}} = \dim \mathcal{S}^h$ is the global number of nodes. Furthermore, $\eta_{g_{\text{i}}} \subset \eta$ refers to the subset of nodes that are prescribed by the Dirichlet data, $u^h_{\text{i}}(x) = g_{\text{i}}(x)$, on $\Gamma_{g_{\text{i}}}$.

Using (17) we can write the strain as,

$$\epsilon(\mathbf{u}) = \sum_{\text{i}=1}^{n_{\text{sd}}} \sum_{A \in \eta} \mathsf{B}_{A\text{i}}(x) \mathrm{u}_{A\text{i}} \tag{23}$$

where,

$$\mathsf{B}_{A1}(x) = \begin{bmatrix} \mathrm{N}_{A,1}(x) \\ 0 \\ 0 \\ 0 \\ \mathrm{N}_{A,3}(x) \\ \mathrm{N}_{A,2}(x) \end{bmatrix} \qquad \mathsf{B}_{A2}(x) = \begin{bmatrix} 0 \\ \mathrm{N}_{A,2}(x) \\ 0 \\ \mathrm{N}_{A,3}(x) \\ 0 \\ \mathrm{N}_{A,1}(x) \end{bmatrix}, \qquad \mathsf{B}_{A3}(x) = \begin{bmatrix} 0 \\ 0 \\ \mathrm{N}_{A,3}(x) \\ \mathrm{N}_{A,2}(x) \\ \mathrm{N}_{A,1}(x) \\ 0 \end{bmatrix}. \tag{24}$$

Alternatively, we write, $\mathsf{B}_{A\text{i}}(x) = \sum_{\text{j}=1}^{n_{\text{sd}}} \mathsf{E}_{:\text{ij}} \mathrm{N}_{A,\text{j}}(x)$,[2] where $\mathsf{E}_{:\text{ij}}$ denotes the ith column vector of the incidence matrix, $\mathsf{E}_{::\text{j}}$, $\text{j} = 1, 2, 3$, given by,

$$\mathsf{E}_{::1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \qquad \mathsf{E}_{::2} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \qquad \mathsf{E}_{::3} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{25}$$

[2] We use Matlab notation to slice multidimensional arrays using the colon operator.

We wish to pull back the above expressions to the parameter domain. Using,

$$\mathrm{N}_{A,\mathrm{j}} = \sum_{\mathrm{k}=1}^{n_{\mathrm{sd}}} \left[DF^{-T}(\hat{x}) \right]_{\mathrm{jk}} \hat{\mathrm{N}}_{A,\mathrm{k}}(\hat{x})$$

we may write,

$$\mathsf{B}_{Ai}(x) = \sum_{\mathrm{k}=1}^{n_{\mathrm{sd}}} \hat{\mathsf{E}}_{:\mathrm{ik}}(\hat{x})\, \hat{\mathrm{N}}_{A,\mathrm{k}}(\hat{x}) \qquad \text{where} \qquad \hat{\mathsf{E}}_{:\mathrm{ik}}(\hat{x}) = \sum_{\mathrm{j}=1}^{n_{\mathrm{sd}}} \mathsf{E}_{:\mathrm{ij}} \left[DF^{-T}(\hat{x}) \right]_{\mathrm{jk}}. \tag{26}$$

Then we can write the contribution to the stiffness matrix corresponding to test function $\mathrm{N}_A(x)$ and trial function $\mathrm{N}_B(x)$ as,

$$\left[\mathsf{K}_{\mathrm{ij}}\right]_{AB} = \int_{\Omega} \mathsf{B}_{A\mathrm{i}}^{T}(x)\, \mathsf{C}(x)\, \mathsf{B}_{B\mathrm{j}}(x)\, d\Omega = \sum_{\mathrm{k}=1}^{n_{\mathrm{sd}}} \sum_{\mathrm{l}=1}^{n_{\mathrm{sd}}} \int_{\hat{\Omega}} \hat{\mathrm{N}}_{A,\mathrm{k}}(\hat{x}) \hat{\mathsf{C}}_{\mathrm{ijkl}}(\hat{x}) \hat{\mathrm{N}}_{B,\mathrm{l}}(\hat{x})\, d\hat{\Omega} \tag{27}$$

where $\hat{\mathsf{C}}_{\mathrm{ijkl}}(\hat{x}) \in \mathbb{R}^{3\times3\times3\times3}$ has components,

$$\left[\hat{\mathsf{C}}_{\mathrm{ij}}(\hat{x})\right]_{\mathrm{kl}} = \hat{\mathsf{E}}_{:\mathrm{ik}}^{T}(\hat{x}) \mathsf{C}(F(\hat{x})) \hat{\mathsf{E}}_{:\mathrm{jl}}(\hat{x}) \det DF(\hat{x}), \quad \mathrm{i,j,k,l} \in \{\, 1, 2, 3 \,\}. \tag{28}$$

To summarize, we have the following matrix problem with $n_{\mathrm{dofs}} = n_{\mathrm{sd}} \cdot n_{\mathrm{nodes}} - \sum_{\mathrm{i}=1}^{n_{\mathrm{sd}}} \dim(\eta_{g_{\mathrm{i}}})$ degrees of freedom:

(M)

Find $\mathbf{d} \in \mathbb{R}^{n_{\mathrm{dofs}}}$ such that,

$$\mathsf{K}\mathbf{d} = \mathbf{f} \tag{29a}$$

with,

$$\mathsf{K} = \left[\begin{array}{c|c|c} \mathsf{K}_{11} & \mathsf{K}_{12} & \mathsf{K}_{13} \\ \hline \mathsf{K}_{21} & \mathsf{K}_{22} & \mathsf{K}_{23} \\ \hline \mathsf{K}_{31} & \mathsf{K}_{32} & \mathsf{K}_{33} \end{array}\right], \qquad \mathbf{d} = \left[\begin{array}{c} \mathbf{d}_1 \\ \hline \mathbf{d}_2 \\ \hline \mathbf{d}_3 \end{array}\right], \qquad \mathbf{f} = \left[\begin{array}{c} \mathbf{f}_1 \\ \hline \mathbf{f}_2 \\ \hline \mathbf{f}_3 \end{array}\right] \tag{29b}$$

where,

$$\left[\mathsf{K}_{\mathrm{ij}}\right]_{AB} = \sum_{\mathrm{k}=1}^{3} \sum_{\mathrm{l}=1}^{3} \int_{\hat{\Omega}} \hat{\mathrm{N}}_{A,\mathrm{k}}(\hat{x}) \hat{\mathsf{C}}_{\mathrm{ijkl}}(\hat{x}) \hat{\mathrm{N}}_{B,\mathrm{l}}(\hat{x})\, d\hat{\Omega} \tag{29c}$$

$$\begin{aligned} [\mathbf{f}_{\mathrm{i}}]_A = & \int_{\hat{\Omega}} \hat{\mathrm{N}}_A(\hat{x}) f_{\mathrm{i}}(\hat{x}) \det DF(\hat{x}) d\hat{\Omega} + \int_{\Gamma_{h_{\mathrm{i}}}} \hat{\mathrm{N}}_A(\hat{x}) h_{\mathrm{i}}(\hat{x}) d\Gamma \\ & - \sum_{\mathrm{k}=1}^{3} \sum_{\mathrm{l}=1}^{3} \int_{\hat{\Omega}} \hat{\mathrm{N}}_{A,\mathrm{k}}(\hat{x}) \hat{\mathsf{C}}_{\mathrm{ijkl}}(\hat{x}) g_{\mathrm{j,l}}^{h}(\hat{x})\, d\hat{\Omega}. \end{aligned} \tag{29d}$$

Keeping i, j fixed, this representation of the bilinear form is equivalent to that in a scalar Poisson problem with a non-isotropic diffusivity law. Consequently, this form of the matrix equations is easier to implement than the form of the matrix equations proposed in [37]. Once the new formation and assembly strategy is applied to a scalar Poisson problem with a non-isotropic diffusivity law, it can be easily extended to linear elasticity.

3.2. Application of sum factorization and weighted quadrature

Sum factorization achieves its efficiency by reordering the operations of a multivariate quadrature rule in such a way that in every stage of the algorithm a univariate quadrature is performed. The essential ingredient is the tensor product structure of both the quadrature rule and the basis functions. In order to exploit this structure, the equations have to be pulled back to the parameter domain. In the previous section we did this in the setting of elastostatics in (29).

We consider application of sum factorization and weighted quadrature to the main term in (29). For a fixed $\mathrm{i,j,k,l} \in 1,2,3$, let $\hat{\mathsf{c}}(\hat{x}) = \hat{\mathsf{C}}_{\mathrm{ijkl}}(\hat{x}) \in \mathbb{R}$. We wish to evaluate the integral,

$$[\mathsf{S}]_{AB} = \int_{\hat{\Omega}} \hat{\mathrm{N}}_{A,\mathrm{k}}(\hat{x})\hat{\mathsf{c}}(\hat{x})\hat{\mathrm{N}}_{B,\mathrm{l}}(\hat{x})\, d\hat{\Omega}. \tag{30}$$

Let $\hat{N}^{(0)}_{a_\mathrm{i}}(\hat{x}_\mathrm{i}) \coloneqq \hat{N}_{a_\mathrm{i}}(\hat{x}_\mathrm{i})$, $\hat{N}^{(1)}_{a_\mathrm{i}}(\hat{x}_\mathrm{i}) \coloneqq \hat{N}'_{a_\mathrm{i}}(\hat{x}_\mathrm{i})$ and let δ_{ki} be equal to 1 if $\mathrm{k} = \mathrm{i}$ and 0 otherwise. Using the tensor product structure of the basis functions (13), we may change the order of integration to obtain,

$$\begin{aligned}[\mathsf{S}]_{AB} = \int_{\hat{\Omega}_1} \hat{N}^{(\delta_{\mathrm{k}1})}_{a_1}(\hat{x}_1)\hat{N}^{(\delta_{\mathrm{l}1})}_{b_1}(\hat{x}_1) \times \Bigg[\dots \\ \int_{\hat{\Omega}_2} \hat{N}^{(\delta_{\mathrm{k}2})}_{a_2}(\hat{x}_2)\hat{N}^{(\delta_{\mathrm{l}2})}_{b_2}(\hat{x}_2) \times \Bigg[\dots \\ \int_{\hat{\Omega}_3} \hat{N}^{(\delta_{\mathrm{k}3})}_{a_3}(\hat{x}_3)\hat{N}^{(\delta_{\mathrm{l}3})}_{b_3}(\hat{x}_3)\, \hat{\mathsf{c}}(\hat{x}_1,\hat{x}_2,\hat{x}_3)\, d\hat{x}_3 \Bigg] d\hat{x}_2 \Bigg] d\hat{x}_1. \end{aligned} \tag{31}$$

Hence, the multivariate integral can be written as a recursion of univariate integrals. The univariate weighted quadrature rules, as defined in Definition 2.1, can now be easily applied to obtain,

$$\begin{aligned}[\mathsf{S}]_{AB} \approx \sum_{i=1}^{q_1} W^{(\delta_{\mathrm{k}1})}_{a_1 i} \hat{N}^{(\delta_{\mathrm{l}1})}_{b_1}(\hat{x}_{1i}) \times \Bigg[\dots \\ \sum_{j=1}^{q_2} W^{(\delta_{\mathrm{k}2})}_{a_2 j} \hat{N}^{(\delta_{\mathrm{l}2})}_{b_2}(\hat{x}_{2j}) \times \Bigg[\dots \\ \sum_{k=1}^{q_3} W^{(\delta_{\mathrm{k}3})}_{a_3 k} \hat{N}^{(\delta_{\mathrm{l}3})}_{b_3}(\hat{x}_{3k})\, \hat{\mathsf{c}}(\hat{x}_{1i},\hat{x}_{2j},\hat{x}_{3k}) \Bigg] \Bigg]. \end{aligned} \tag{32}$$

In practice, the integral is evaluated as follows. Input is a three-dimensional array,

$$\{\, \hat{\mathsf{c}}(\hat{x}_{1i},\hat{x}_{2j},\hat{x}_{3k}),\ i \in 1:q_1,\ j \in 1:q_2,\ k \in 1:q_3 \,\}. \tag{33}$$

In the inner stage of the recursion this array is contracted along one dimension to a matrix. This matrix is subsequently contracted to a vector in the second stage of the recursion. Finally, in the outer stage, the vector is reduced to a single number that is the approximation of the integral corresponding to index AB. The other terms in (29) can similarly be evaluated using sum factorization and weighted quadrature.

4. Implementational aspects of the methodology

In this section we discuss several important implementational aspects of the methodology. Firstly, we discuss efficient implementation of sum factorization and provide useful pseudocode. The implementation does not depend on the type of assembly method nor on the type of quadrature rule. Furthermore, we discuss efficient access and assignment into the prevalent sparse matrix data structures, namely, Compressed Sparse Row (CSR) and Compressed Sparse Column (CSC). In particular, row-by-row or column-by-column assembly allows matrix rows or columns, respectively, to be formed contiguously in the storage order of the sparse matrix, thereby minimizing the memory overhead and eliminating the addition assignment operation on sparse matrices. Finally we discuss opportunities for shared and distributed memory parallelism.

4.1. Efficient implementation of sum factorization

Sum factorization is performed in several stages. In a three-dimensional problem the input is a five-dimensional array,[3] $\mathsf{C} \in \mathbb{R}^{q_1 \times q_2 \times q_3 \times 1 \times 1}$, where q_i represents the number of quadrature points in component direction i. The entries of this array encode the metric and material dependence evaluated at the quadrature points. In each one of

[3] Note that C is in fact a three-dimensional array because of the two trailing singleton dimensions. These extra dimensions are convenient in template programming such that in each stage of the sum factorization the dimension of the input array reduces by one dimension.

the three stages of the sum factorization algorithm the input array is contracted along one dimension. The final result will be a matrix.

Algorithm 1 provides pseudocode for the efficient implementation of sum factorization. The function *kron* represents the standard Kronecker product of two matrices. The loops are nested in such a way that entries in arrays A and C are accessed and assigned to in storage order.

The implementation does not depend on the chosen assembly method, that is, it works in the case of element-by-element, row-by-row or column-by-column formation and assembly, nor does it depend on the chosen quadrature rule.[4] The specific dimension of the input arrays does depend on the chosen assembly method:

- Element-by-element assembly: $n_i,\ m_i > 1,\ i = 1, 2, 3.$
- Row-by-row assembly: $n_i > 1,\ m_i = 1,\ i = 1, 2, 3.$
- Column-by-column assembly: $n_i = 1,\ m_i > 1,\ i = 1, 2, 3.$

```
Function sumfact_stage_1 (C, testfuns, trialfuns)
    input : C ∈ ℝ^{q1×q2×q3×1×1}, testfuns ∈ ℝ^{n3×q3}, trialfuns ∈ ℝ^{m3×q3}
    output: A ∈ ℝ^{n3×m3×q1×q2}
    Loop over 3rd array dimension
    for k ← 1 to q3 do
        B = testfuns(:, k) ∗ trialfuns(:, k)^T
        Loop over 2nd array dimension
        for j ← 1 to q2 do
            Loop over 1st array dimension
            for i ← 1 to q1 do
                A(:, :, i, j) += C(i, j, k, 1, 1) ∗ B
            end
        end
    end
end
```

```
Function sumfact_stage_2 (C, testfuns, trialfuns)
    input : C ∈ ℝ^{n3×m3×q1×q2}, testfuns ∈ ℝ^{n2×q2}, trialfuns ∈ ℝ^{m2×q2}
    output: A ∈ ℝ^{(n3·n2)×(m3·m2)×q1}
    Loop over 2nd array dimension
    for j ← 1 to q2 do
        B = testfuns(:, j) ∗ trialfuns(:, j)^T
        Loop over 1st array dimension
        for i ← 1 to q1 do
            A(:, :, i) += kron (C(:, :, i, j), B)
        end
    end
end
```

Table 3 summarizes the number of floating point operations of the algorithm that has been used to estimate the computational cost in Table 1. As discussed in the introduction, combined with a traditional element loop, sum factorization reduces the number of FLOPS from $\mathcal{O}(p^9)$ to $\mathcal{O}(p^7)$. Combined with a row or column loop and in weighted quadrature the complexity is further reduced by a factor of p^3 resulting in a method that scales $\mathcal{O}(p^4)$ per degree of freedom.

Sum factorization significantly speeds up the quadrature process. However, besides the implementation burden, the main drawback of sum factorization is that the metric and material dependent part needs to be precomputed at

[4] We assume that the quadrature weights have been incorporated in `testfuns`.

Function *sumfact_stage_3* (C, *testfuns*, *trialfuns*)
  input : $\mathsf{C} \in \mathbb{R}^{(n_3 \cdot n_2) \times (m_3 \cdot m_2) \times q_1}$, $\mathtt{testfuns} \in \mathbb{R}^{n_1 \times q_1}$, $\mathtt{trialfuns} \in \mathbb{R}^{m_1 \times q_1}$
  output: $\mathsf{A} \in \mathbb{R}^{(n_3 \cdot n_2 \cdot n_1) \times (m_3 \cdot m_2 \cdot m_1)}$
  Loop over 1st array dimension
  for $i \leftarrow 1$ **to** q_1 **do**
    $\mathsf{B} = \mathtt{testfuns}(:, i) * \mathtt{trialfuns}(:, i)^T$
    $\mathsf{A}(:, :) \mathrel{+}= \mathrm{kron}\left(\mathsf{C}(:, :, i), \mathsf{B}\right)$
  end
end

input : $\mathsf{C} \in \mathbb{R}^{q_1 \times q_2 \times q_3 \times 1 \times 1}$, $\mathtt{test} \in \left(\mathbb{R}^{n_1 \times q_1}, \mathbb{R}^{n_2 \times q_2}, \mathbb{R}^{n_3 \times q_3}\right)$, $\mathtt{trial} \in \left(\mathbb{R}^{m_1 \times q_1}, \mathbb{R}^{m_2 \times q_2}, \mathbb{R}^{m_3 \times q_3}\right)$
output: $\mathsf{C} \in \mathbb{R}^{(n_3 \cdot n_2 \cdot n_1) \times (m_3 \cdot m_2 \cdot m_1)}$
Loop over all 3 array dimensions
$\mathsf{C} = \mathrm{sumfact\_stage\_1}\left(\mathsf{C}, \mathtt{test}\{3\}, \mathtt{trial}\{3\}\right)$
$\mathsf{C} = \mathrm{sumfact\_stage\_2}\left(\mathsf{C}, \mathtt{test}\{2\}, \mathtt{trial}\{2\}\right)$
$\mathsf{C} = \mathrm{sumfact\_stage\_3}\left(\mathsf{C}, \mathtt{test}\{1\}, \mathtt{trial}\{1\}\right)$

Algorithm 1: Sum factorization. n_i and m_i represent the number of test and trial functions, respectively, that are integrated in component direction i. q_i represents the number of quadrature evaluations in component direction i. We note that in row formation $n_\mathrm{i} = 1,\ m_\mathrm{i} > 1$, in column formation $m_\mathrm{i} = 1, n_\mathrm{i} > 1$ and in element-by-element formation $n_\mathrm{i}, m_\mathrm{i} > 1$. The indexing into the multidimensional array C is such that computations are performed contiguously in the storage order of the array. Finally, it is assumed that the quadrature weights are combined with the test function array.

Table 3
Floating point operations of sum factorization.

	Stage 1	Stage 2	Stage 3
FLOPS	$2(n_3 \cdot m_3) \cdot q_1 \cdot q_2 \cdot q_3$	$2(n_3 \cdot m_3) \cdot (n_2 \cdot m_2) \cdot q_1 \cdot q_2$	$2(n_3 \cdot m_3) \cdot (n_2 \cdot m_2) \cdot (n_2 \cdot m_2) \cdot q_1$

the quadrature points. In element-by-element assembly this is not an issue because the array sizes remain constant and relatively small ($(p+1)^{n_\mathrm{sd}}$ points per element). However, the presented weighted quadrature scheme operates patch wise and the number of quadrature points basically has no upper bound with patch refinement. Therefore, a serious implementation of these ideas needs some form of domain or matrix partitioning and load balancing. This can be directly combined with parallel computation of the matrix rows or columns.

4.2. Sparse matrix data structures

There exist different data structures for storing sparse data in a matrix. There are storage formats that allow for efficient access and ease of handling and formats that allow for efficient storage and matrix operations. We are interested in the second group, because those are the sparse matrix data structures typically encountered in scientific computing. In particular, we are interested in the following two sparse matrix formats:

(i) Compressed Sparse Row (CSR).

(ii) Compressed Sparse Column (CSC).

The CSC data structure is supported by scientific languages such as Matlab and Julia, the CSR format is used in high performance packages such as Trilinos and PETSc, and some scientific packages, such as ScyPy for the Python language, support both.

Let n_{nz} denote the number of nonzero entries of a sparse matrix A. Its compressed sparse row format is represented by the triple (`rowptr`, `colind`, `nzval`). Here `nzval` contains the n_{nz}-nonzero values of A in *row-major*

Table 4
CSR format of A.

	0	1	2	3	4	5	6	7	8
`nzval`	11	22	33	44	55	66	77	88	99
`colind`	0	2	0	3	4	1	2	1	4
`rowptr`	0	2	5	7	9				

Table 5
CSC format of A.

	0	1	2	3	4	5	6	7	8
`nzval`	11	33	66	88	22	77	44	55	99
`rowind`	0	1	2	3	0	2	1	1	3
`colptr`	0	2	4	6	7	9			

ordering, i.e. ordered from left to right and from top to bottom, `colind` stores the column index of each value in `nzval`, and `rowptr` is the list of `nzval`-indices that start a new row in A.

The CSC storage format is similar to the CSR format. Instead, the n_{nz}-nonzero values of A are stored in column major ordering in the vector `nzval`, row indices are stored in `rowind` and pointers to the first nonzero value in each column of A are stored in `colptr`. Example 4.2 depicts an example of a matrix in CSR and CSC format, respectively.

Remark 4.1. One can readily check that the CSC representation of matrix A is equal to the CSR format of A^T.

Example 4.2 (*Sparse matrix data structures*)**.** Consider the sparse matrix,

$$\mathsf{A} = \begin{bmatrix} 11 & 0 & 22 & 0 & 0 \\ 33 & 0 & 0 & 44 & 55 \\ 0 & 66 & 77 & 0 & 0 \\ 0 & 88 & 0 & 0 & 99 \end{bmatrix}, \tag{34}$$

(i) The CSR format of A is shown in Table 4

(ii) The CSC format of A is shown in Table 5

Note, we have used zero-based indexing because it is usually preferred over 1-based indexing.

As it turns out, the row-by-row finite element formation and assembly is tailored to the CSR structure. This is due to the fact that the entries of the matrix are computed in the same order as the row-major ordering of the CSR matrix. On the other hand, column-by-column formation and assembly is more efficient in combination with the CSC storage format.

4.3. Domain partitioning and parallel formation and assembly

Row or column formation and assembly shares the beneficial properties of the interaction-wise approach proposed in [42] — in which matrix entries are computed one-by-one — regarding access and assignment of sparse matrices and opportunities for shared and distributed memory parallelism. Consider the matrix equations in (29). We list the following:

1. The nine stiffness matrices K_{ij} corresponding to $i, j \in \{ 1, 2, 3 \}$ can be independently formed. Hence, computing these matrices can be done efficiently in parallel on a shared memory or distributed memory parallel machine.
2. Keeping i, j fixed, the matrices corresponding to $k, l \in \{ 1, 2, 3 \}$ can be formed by a parallel reduction, which is attuned to shared memory parallelism.

3. On a shared memory machine the rows or columns of K_{ij} $i, j \in \{ 1, 2, 3 \}$ can be independently formed in parallel.
4. If one were to recompute the metric and material dependent data for each test or trial function separately then rows or columns could be formed independently in parallel on a shared or distributed parallel machine.

Although the last approach leads to many more evaluations of the metric and constitutive equations it leads to an approach that is both simple and embarrassingly parallel. In fact, it can be considered as a domain partitioning procedure, in which the support of a single test or trial function is the relevant domain that follows from the partitioning. If we assume that the load is uniform across the rows or columns of the matrix, then this leads to a very simple domain decomposition and load balancing scheme.

5. Numerical results

In this section we present results for several numerical benchmark problems that illustrate the efficiency and efficacy of the proposed formation and assembly techniques applied in the context of isogeometric linear elasticity. In particular, we show that the accuracy of full Gauss quadrature is maintained while the computational burden of forming the matrix equations is significantly reduced. The following benchmark problems are considered:

1. Hole in plate problem (2D smooth solution solved as a 3D problem).
2. Spherical cavity problem (3D smooth solution).

In every benchmark problem we compare the accuracy of weighted quadrature versus full Gauss quadrature by investigating the $L^2(\Omega)$ error in displacement and stress. Full Gauss quadrature requires $(p+1)^{n_{\text{sd}}}$ points per element compared to roughly $3^{n_{\text{sd}}}$ points per element, measured asymptotically with k-refinement, for weighted quadrature.

5.1. Setup

The ideas in this paper have been implemented in the Julia language [50]. Julia features the Compressed Sparse Column (CSC) storage format for sparse matrices, which is why we employ a column loop instead of a row loop. The performance of the new methodology is studied by comparing absolute and relative timings of the following combinations of formation and assembly procedures:

1. Element loop with standard quadrature loop employing full Gauss quadrature.
2. Element loop with standard quadrature loop employing weighted quadrature.
3. Element loop with sum factorization employing full Gauss quadrature.
4. Element loop with sum factorization employing weighted quadrature.
5. Column loop with sum factorization employing weighted quadrature.

All combinations of techniques employ the same representation of the matrix equations of linear elasticity (29), use the same NURBS description of the geometry and the same test and trial spaces based on B-splines. We therefore consider this as a fair comparison. Note that we no longer strictly conform to the isoparametric concept that is usually applied in isogeometric analysis, because the solution variables are represented in terms of B-splines while the geometry is represented by NURBS.

The absolute timings are obtained on a 2.7 GHz E5 − 2680 (Sandy Bridge) processor with 32 GB RAM. We measure only the formation and assembly of the stiffness matrix and right-hand-side forcing vector. The timings do not include the solution of the linear system using MUMPS, a high performance direct solver. Finally, the absolute timings are normalized by the presented approach (# 5) in this paper, that implements a column loop with sum factorization employing weighted quadrature. These results give an indication of the speedup that can be expected for three-dimensional problems of engineering interest.

5.2. Infinite plate with circular hole under constant in-plane tension

Consider a thick infinite plate with a circular hole under constant in-plane tension at infinity [51,52]. The infinite plate is modeled by a finite quarter plate using symmetry conditions as depicted in Fig. 4 and plane strain

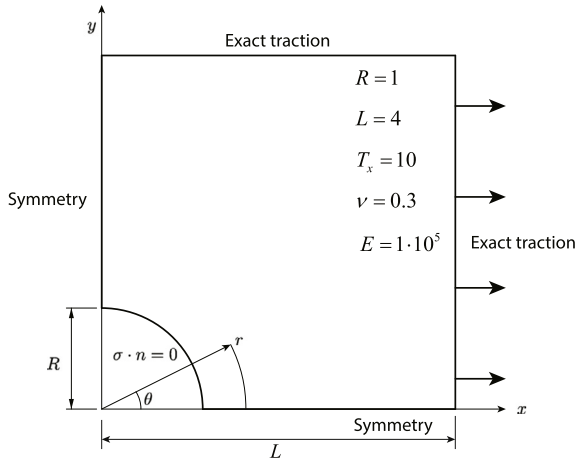


Fig. 4. Elastic plate with a circular hole: problem definition.

displacement conditions. The exact solution of displacement and stress, represented in polar coordinates (r, θ), is,

$$u_r(r,\theta) = \frac{\nu+1}{E}\left(\frac{rT_x}{4}\left(\frac{2R^2}{r^2}+2-4\nu\right)+\frac{rT_x\cos(2\theta)}{2}\left(-\frac{R^4}{r^4}+(4-4\nu)\frac{R^2}{r^2}+1\right)\right)$$

$$u_\theta(r,\theta) = -\frac{rT_x\sin(2\theta)(\nu+1)}{2E}\left(\frac{R^4}{r^4}+(2-4\nu)\frac{R^2}{r^2}+1\right)$$

and,

$$\sigma_{rr}(r,\theta) = \frac{T_x}{2}\left(1-\frac{R^2}{r^2}\right)+\frac{T_x}{2}\left(1-4\frac{R^2}{r^2}+3\frac{R^4}{r^4}\right)\cos(2\theta)$$

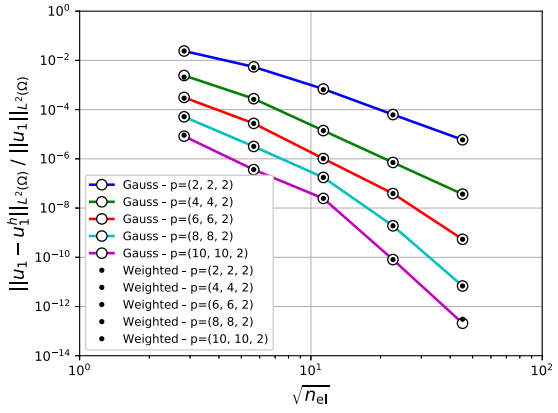
$$\sigma_{\theta\theta}(r,\theta) = \frac{T_x}{2}\left(1+\frac{R^2}{r^2}\right)-\frac{T_x}{2}\left(1+3\frac{R^4}{r^4}\right)\cos(2\theta)$$

$$\sigma_{r\theta}(r,\theta) = -\frac{T_x}{2}\left(1+2\frac{R^2}{r^2}-3\frac{R^4}{r^4}\right)\sin(2\theta).$$

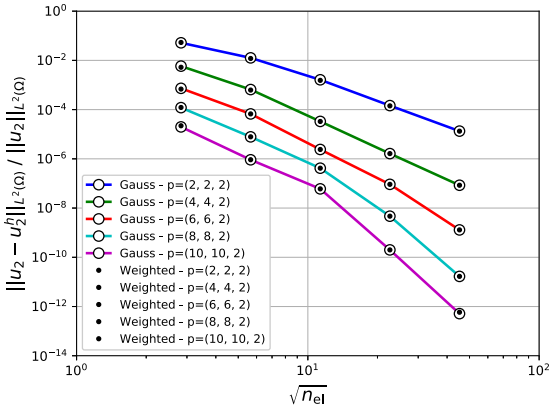
Here T_x is the applied force at infinity and R is the radius of the hole in the quarter plate. Furthermore, E and ν are the Young's modulus and Poisson's ratio of the isotropic linearly elastic material.

Although the problem definition is two-dimensional, we model the plate using a single three dimensional quadratic NURBS patch with one quadratic element through the thickness. The knot vectors of the coarsest geometry description are $\Xi_1 = \{\,0,0,0,1,1,1\,\}$, $\Xi_2 = \{\,0,0,0,1,2,2,2\,\}$ and $\Xi_3 = \{\,0,0,0,1,1,1\,\}$. Note that there is one internal knot in the second parametric direction. Under p- or k-refinement the smoothness of the test and trial spaces must remain C^1 at this knot. The upper-right corner is created by collapsing edges of the control mesh, creating a singular edge. We note that the Jacobian and its determinant are undefined there. This is not an issue because we have no quadrature points that coincide with this singularity. We refer to [5] for a complete description of the geometric model. The conditions of plain strain, as well as the in-plane symmetry conditions, are imposed strongly. The exact tractions are imposed weakly.

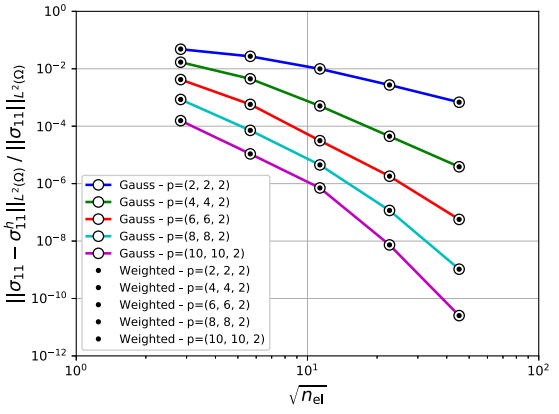
In Fig. 5 the accuracy of weighted quadrature is verified by comparing directly with numerical benchmark results obtained using full Gauss quadrature. The $L^2(\Omega)$ errors in displacement and stress are compared under uniform h-refinement for polynomial degrees $(p, p, 2)$, $p = 2, 4, 6, 8$ and 10. Virtually no loss in accuracy is observed when using weighted quadrature versus full Gauss quadrature in both displacement and stress. The absolute and relative timings depicted in Fig. 6 report speedups by a factor of 10 at low p and up to 500 at high p, compared to traditional formation and assembly techniques. Compared to the fastest existing finite element procedures employing sum factorization in an element loop, the new procedures, employing a column loop with sum factorization and weighted quadrature, are a factor of 3–20 faster.



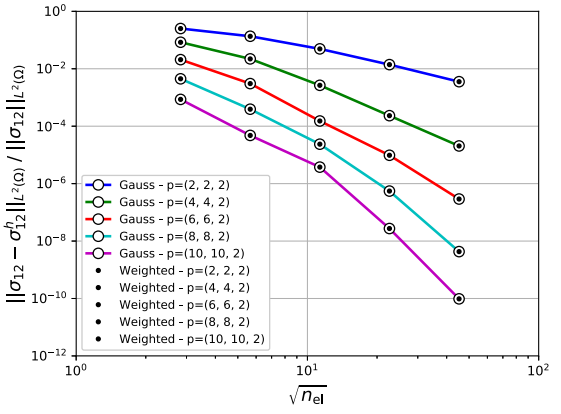
(a) Relative L^2 error in u_1.



(b) Relative L^2 error in u_2.

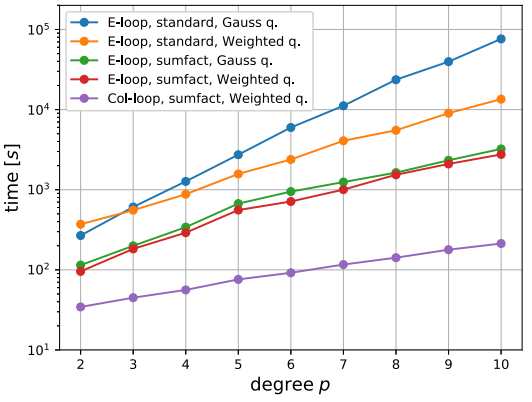


(c) Relative L^2 error in σ_{11}.

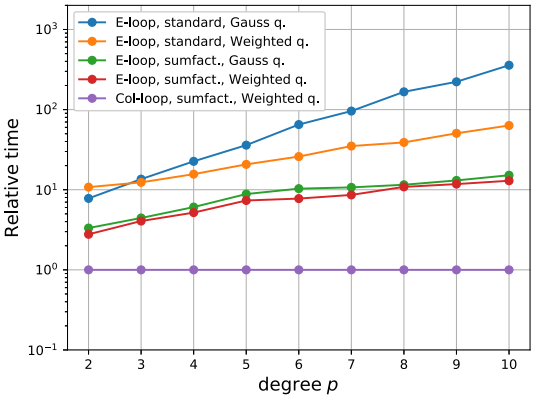


(d) Relative L^2 error in σ_{12}.

Fig. 5. Comparison of the accuracy of weighted quadrature versus standard Gauss quadrature for the hole in plate problem.



(a) Absolute formation and assembly times.



(b) Assembly time relative to col-loop with sum factorization and weighted quadrature.

Fig. 6. Absolute and relative formation and assembly timings for different formation and assembly strategies applied to the hole in plate problem. The results are obtained using a $32 \times 64 \times 1$ element mesh of polynomial degree $(p, p, 2)$.

5.3. Spherical cavity in an infinite medium subjected to uniform tension at infinity

The second benchmark problem demonstrates the performance of the proposed methodology for three-dimensional spline discretizations. Consider an infinite three-dimensional medium with a spherical cavity located at the origin and uniform uniaxial tension in the z-direction at infinity, see Fig. 7. This is a classical problem in isotropic linear elasticity and has the following analytical solution [51,52][5] in terms of displacement and stress in spherical coordinates (r, β, θ),

$$
\begin{aligned}
u_r =& \frac{3R^5T_z}{2(7-5\nu)r^4} - \frac{R^3(6-5\nu)T_z}{2(7-5\nu)r^2} + -\frac{\nu r T_z}{\nu+1} \\
& \cos^2(\beta)\left(-\frac{9R^5T_z}{2(7-5\nu)r^4} - \frac{5R^3(4\nu-5)T_z}{2(7-5\nu)r^2} + \frac{3\nu r T_z}{\nu+1} - \frac{(4\nu-2)rT_z}{2(\nu+1)}\right) \\
u_\beta =& \sin(\beta)\cos(\beta)\left(-\frac{3R^5T_z}{(7-5\nu)r^4} - \frac{5R^3T_z}{2(7-5\nu)r^2} - \frac{3\nu r T_z}{\nu+1} - \frac{rT_z}{2(\nu+1)}\right) \\
u_\theta =& \, 0
\end{aligned}
$$

and,

$$
\begin{aligned}
\sigma_{rr} &= T_z\cos(\beta)^2 + \frac{T_z}{7-5\nu}\left(\frac{R^3}{r^3}(6-5(5-\nu)\cos(\beta)^2) + \frac{6R^5}{r^5}(3\cos(\beta)^2-1)\right) \\
\sigma_{\theta\theta} &= \frac{3T_z}{2(7-5\nu)}\left(\frac{R^3}{r^3}(5\nu-2+5(1-2\nu)\cos(\beta)^2) + \frac{R^5}{r^5}(1-5\cos(\beta)^2)\right) \\
\sigma_{\beta\beta} &= T_z\sin(\beta)^2 + \frac{T_z}{2(7-5\nu)}\left(\frac{R^3}{r^3}(4-5\nu+5(1-2\nu)\cos(\beta)^2) + \frac{3R^5}{r^5}(3-7\cos(\beta)^2)\right) \\
\sigma_{r\beta} &= T_z\left(-1+\frac{1}{7-5\nu}\left(-\frac{5R^3}{r^3}(1+\nu)+\frac{12R^5}{r^5}\right)\right)\sin(\beta)\cos(\beta).
\end{aligned}
$$

Here T_z is the applied force at infinity and R is the inner radius of the spherical cavity.

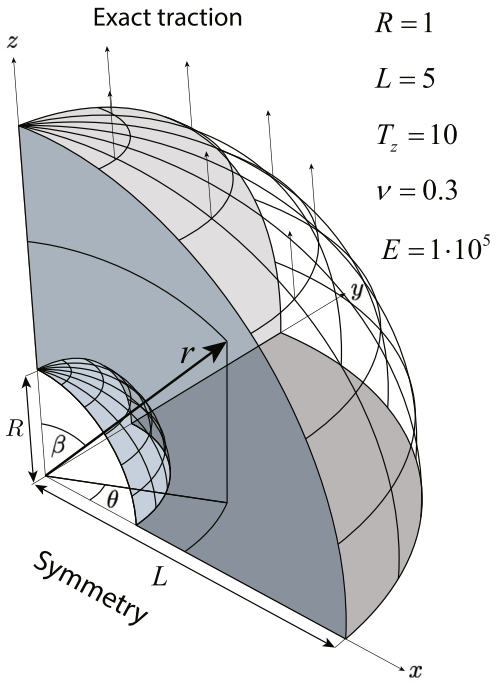
We use the symmetry of the analytical solution, and model the geometry using a single tri-quadratic NURBS patch with knot vectors $\Xi_1 = \Xi_2 = \Xi_3 = \{\, 0, 0, 0, 1, 1, 1 \,\}$. The spherical geometry is modeled exactly by collapsing one of the faces to an edge. This creates a singular line that coincides with the z-axis. Similar to the previous problem this is not an issue because there are no quadrature points located along this edge.

The accuracy of weighted quadrature is studied by comparing the h-convergence behavior with results obtained using full Gauss integration. This is done for polynomial degrees $2, 4, 6$ and 8, see Fig. 8. As in the previous benchmark test case, weighted quadrature attains the same accuracy as full Gauss quadrature for both displacement and stress. In Fig. 9a the absolute time to form the isogeometric stiffness matrix using each of the five different formation and assembly procedures is measured for a fixed $16 \times 16 \times 16$ mesh with polynomial degrees ranging from 2 to 8. Fig. 9b illustrates the speed-ups attainable when using a row or column loop with sum factorization and weighted quadrature. We report speedups compared to traditional approaches ranging from a factor of 18 at low polynomial degree to well over a 1000 at polynomial degree 8. Compared to the fastest existing finite element procedures employing sum factorization, we achieve speed-ups ranging from a factor of 8 at low polynomial degree $p = 2$ up to a factor of 40 at polynomial degree equal to $p = 8$.

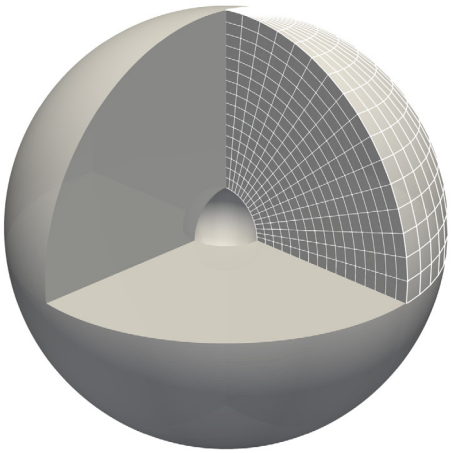
6. Conclusion

This work continues the study in [1] which proposed a novel formation and assembly strategy for finite element analysis that attains significant speedups compared to existing methods while maintaining the accuracy of the Galerkin method. The novel methodology relies on three key ingredients: (1) assembly row-by-row or column-by-column, instead of element-by-element; and an efficient formation strategy based on (2) sum factorization and (3) weighted quadrature, that is applied to each specific row or column of the matrix. This unique selection of

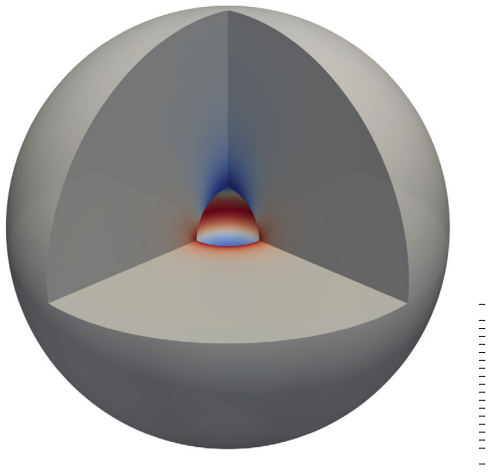
[5] Several textbooks and publications contain errors in the worked out solution to this problem. [51,53] contain worked out solutions to the stress which are not correct. At http://www-personal.umich.edu/~jbarber/elasticity/errata.pdf a correction can be found accompanying the book [51]. [54] contains an analytical solution to the displacement and stress, the former which is correct and the latter which is not.



(a) Problem definition.



(b) $16 \times 16 \times 16$ mesh.

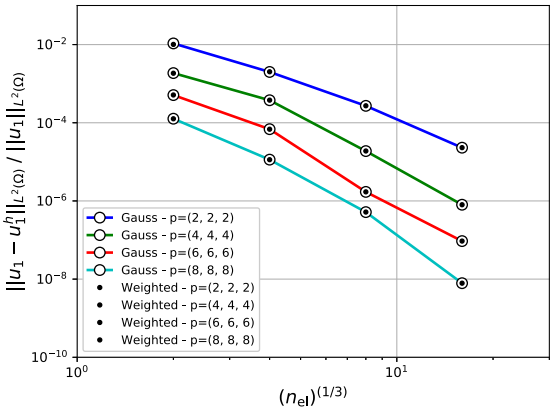


(c) Stress component σ_{zz}.

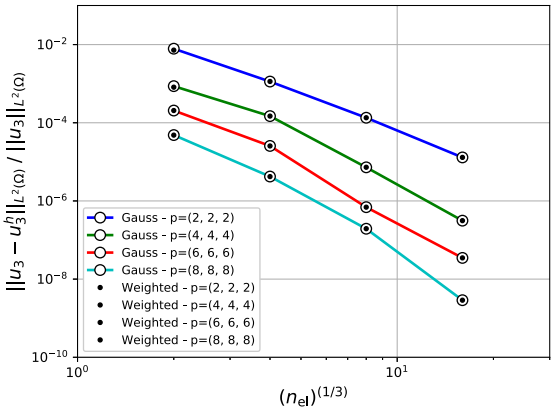
Fig. 7. Infinite medium with spherical hole in uniaxial tension: (a) Problem definition; (b) Isogeometric mesh; and (c) Analytical stress component in z-direction.

techniques leads to a formation and assembly approach that scales favorably with polynomial degree p, opening the door for higher-order isogeometric analysis employing k-refinement. For three-dimensional problems, a rough count of the number of floating point operations reveals that,

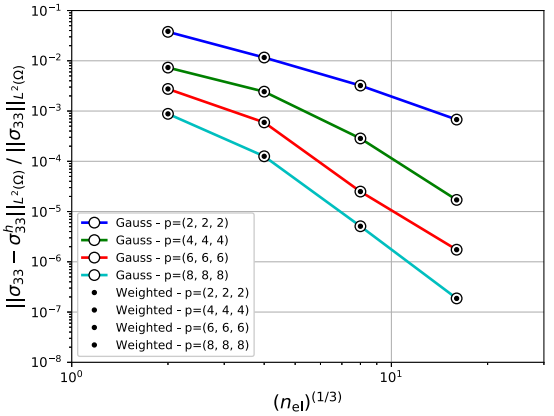
1. Sum factorization lowers the complexity from $\mathcal{O}(p^9)$ per degree of freedom to $\mathcal{O}(p^7)$.
2. Weighted quadrature rules scale optimally with the polynomial degree, that is the number of quadrature points does not depend on p, which reduces the computational burden further down to $\mathcal{O}(p^6)$ per degree of freedom.



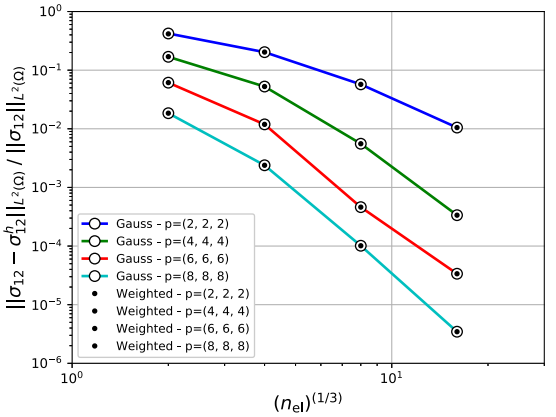
(a) Relative L^2 error in u_1.



(b) Relative L^2 error in u_3.

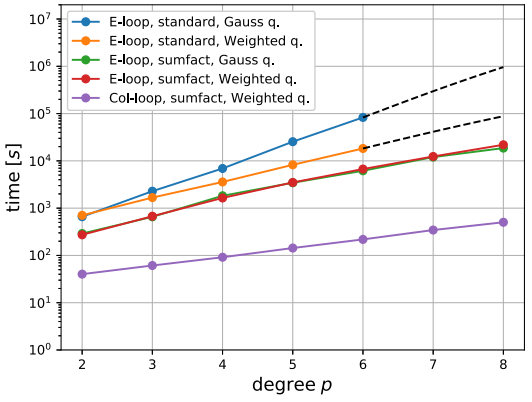


(c) Relative L^2 error in σ_{33}.

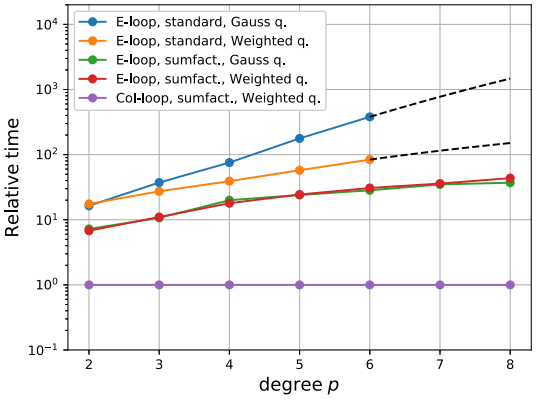


(d) Relative L^2 error in σ_{12}.

Fig. 8. Comparison of the accuracy of weighted quadrature versus standard Gauss quadrature for the spherical cavity problem.



(a) Absolute formation and assembly times.



(b) Assembly time relative to col-loop with sum factorization and weighted quadrature.

Fig. 9. Absolute and relative formation and assembly times for different formation and assembly strategies applied to the spherical cavity problem with uniform open knot vectors with $16 \times 16 \times 16$ elements of uniform polynomial degree p. The results associated with a standard quadrature loop have been extrapolated based on their FLOPS count of $\mathcal{O}(p^9)$ and $\mathcal{O}(p^6)$.

3. Sum factorization combined with weighted quadrature in a row or column assembly reduces the complexity further to $\mathcal{O}(p^4)$ per degree of freedom, which is close to the theoretical optimum $\mathcal{O}(p^3)$ obtained by collocation.

In this work we have made several contributions, with a particular focus on efficiency, robustness and generality, while maintaining the accuracy of the Galerkin method. Specifically, we have extended the weighted quadrature scheme to accurately integrate the entries of the stiffness matrix in linear elasticity. We proposed a new distribution of quadrature points that works in the general setting of non-uniform, mixed continuity isogeometric spaces. Furthermore, we investigated efficient access and assignment into the prevalent sparse matrix data structures, namely, Compressed Sparse Row (CSR) and Compressed Sparse Column (CSC).

Numerical studies verified the gain in efficiency of the methodology compared to existing techniques. In each case, the accuracy of the Galerkin method was maintained while significantly reducing the computational time. Compared to traditional finite element formation and assembly techniques, using full Gauss–Legendre quadrature, we reduced the time by a factor of 18 for $p = 2$, a factor of 80 for $p = 4$ and a factor of well over a thousand for polynomial degree $p = 8$. Compared to the fastest element-by-element assembly techniques, employing sum factorization, we have reported savings up to a factor of 8 for $p = 2$, a factor of 18 for $p = 4$, up to a factor of 40 for $p = 8$.

Although the advantages of the methodology are clear there are several disadvantages of the approach. Firstly, finite element routines will have to be completely rewritten. This could be done in a step by step fashion. First, sum factorization and weighted quadrature can be combined within a standard element loop. Once those facets of the methodology are in place, one could change to a row or column loop. Another disadvantage of the formation and assembly strategy is that many quantities need to be precomputed for use in sum factorization. One could consider to recomputing the geometry and constitutive data on the fly. This way, rows or columns can be formed independently, resulting in a very scalable formation and assembly. Evaluation of constitutive equations can, however, be very expensive for certain types of physical phenomena. Hence, it will always be a trade-off of minimizing the total number of evaluations versus minimizing memory use.

In future work we will explore weighted quadrature in the setting of non-uniform spline spaces with a focus on stability and accuracy. Also, we plan to extend the methodology to unstructured discretizations employing local refinement through truncated hierarchical B-splines, proceeding along the lines of [33].

Acknowledgments

T.J.R. Hughes and R.R. Hiemstra were partially supported by the Office of Naval Research, United States (Grant Nos. N00014-17-1-2119, N00014-17-1-2039, and N00014-13-1-0500), by the Army Research Office, United States (Grant No. W911NF-13-1-0220), by the National Institutes of Health, United States (NIH Project 5R01HL129077-02,03), and by the National Science Foundation Industry/University Cooperative Research Center (IUCRC) for Efficient Vehicles and Sustainable Transportation Systems (EV-STS), United States, and the United States Army CCDC Ground Vehicle Systems Center (TARDEC/NSF Project # 1650483 AMD 2). This support is gratefully acknowledged. G. Sangalli was partially supported by the European Research Council through the FP7 Ideas Consolidator Grant *HIGEOM* n. 616563, and by the Italian Ministry of Education, University and Research (MIUR) through the "Dipartimenti di Eccellenza Program (2018–2022) - Dept. of Mathematics, University of Pavia". This support is gratefully acknowledged. F. Calabrò, G. Sangalli and M. Tani are members of the Gruppo Nazionale Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM). The INdAM support through GNCS "Progetti di Ricerca 2018" and "Progetti giovani ricercatori 2018" programs is gratefully acknowledged.

References

[1] F. Calabrò, G. Sangalli, M. Tani, Fast formation of isogeometric Galerkin matrices by weighted quadrature, Comput. Methods Appl. Mech. Engrg. 316 (2017) 606–622.

[2] A. Buffa, J. Rivas, G. Sangalli, R. Vázquez, Isogeometric discrete differential forms in three dimensions, SIAM J. Numer. Anal. 49 (2) (2011) 818–844.

[3] T.J.R. Hughes, The Finite Element Method: Linear Static and Dynamic Finite Element Analysis, Courier Corporation, 2012.

[4] G. Strang, G.J. Fix, An Analysis of the Finite Element Method, vol. 212, Prentice-Hall Englewood Cliffs, NJ, 1973.

[5] J.A. Cottrell, T.J.R. Hughes, Y. Bazilevs, Isogeometric Analysis: Toward Integration of CAD and FEA, John Wiley & Sons, 2009.

[6] T.J.R. Hughes, J.A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Comput. Methods Appl. Mech. Engrg. 194 (39) (2005) 4135–4195.
[7] Y. Bazilevs, L. Beirão da Veiga, J.A. Cottrell, T.J.R. Hughes, G. Sangalli, Isogeometric analysis: approximation, stability and error estimates for h-refined meshes, Math. Models Methods Appl. Sci. 16 (07) (2006) 1031–1090.
[8] S. Lipton, J.A. Evans, Y. Bazilevs, T. Elguedj, T.J.R. Hughes, Robustness of isogeometric structural discretizations under severe mesh distortion, Comput. Methods Appl. Mech. Engrg. 199 (5) (2010) 357–373.
[9] J.A. Cottrell, T.J. Hughes, A. Reali, Studies of refinement and continuity in isogeometric structural analysis, Comput. Methods Appl. Mech. Engrg. 196 (41) (2007) 4160–4183.
[10] J.A. Evans, T.J.R. Hughes, Discrete spectrum analyses for various mixed discretizations of the Stokes eigenproblem, Comput. Mech. 50 (6) (2012) 667–674.
[11] T.J.R. Hughes, J.A. Evans, A. Reali, Finite element and NURBS approximations of eigenvalue, boundary-value, and initial-value problems, Comput. Methods Appl. Mech. Engrg. 272 (2014) 290–320.
[12] T.J.R. Hughes, A. Reali, G. Sangalli, Duality and unified analysis of discrete approximations in structural dynamics and wave propagation: comparison of p-method finite elements with k-method NURBS, Comput. Methods Appl. Mech. Engrg. 197 (49) (2008) 4104–4124.
[13] I. Akkerman, Y. Bazilevs, V.M. Calo, T.J.R. Hughes, S. Hulshoff, The role of continuity in residual-based variational multiscale modeling of turbulence, Comput. Mech. 41 (3) (2008) 371–378.
[14] Y. Bazilevs, V.M. Calo, J.A. Cottrell, T.J.R. Hughes, A. Reali, G. Scovazzi, Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows, Comput. Methods Appl. Mech. Engrg. 197 (1–4) (2007) 173–201.
[15] J.A. Evans, T.J.R. Hughes, Isogeometric divergence-conforming B-splines for the unsteady Navier–Stokes equations, J. Comput. Phys. 241 (2013) 141–167.
[16] R.R. Hiemstra, D. Toshniwal, R.H.M. Huijsmans, M.I. Gerritsma, High order geometric methods with exact conservation properties, J. Comput. Phys. 257 (2014) 1444–1471.
[17] M.J. Borden, M.A. Scott, J.A. Evans, T.J.R. Hughes, Isogeometric finite element data structures based on Bézier extraction of nurbs, Internat. J. Numer. Methods Engrg. 87 (1–5) (2011) 15–47.
[18] M.A. Scott, M.J. Borden, C.V. Verhoosel, T.W. Sederberg, T.J.R. Hughes, Isogeometric finite element data structures based on Bézier extraction of T-splines, Internat. J. Numer. Methods Engrg. 88 (2) (2011) 126–156.
[19] D.J. Benson, Y. Bazilevs, M.C. Hsu, T.J.R. Hughes, Isogeometric shell analysis: the reissner–mindlin shell, Comput. Methods Appl. Mech. Engrg. 199 (5–8) (2010) 276–289.
[20] S. Morganti, F. Auricchio, D.J. Benson, F.I. Gambarin, S. Hartmann, T.J.R. Hughes, A. Reali, Patient-specific isogeometric structural analysis of aortic valve closure, Comput. Methods Appl. Mech. Engrg. 284 (2015) 508–520.
[21] F. Auricchio, L. Beirão da Veiga, T.J.R. Hughes, A. Reali, G. Sangalli, Isogeometric collocation methods, Math. Models Methods Appl. Sci. 20 (11) (2010) 2075–2107.
[22] F. Auricchio, L. Beirão da Veiga, T.J.R. Hughes, A. Reali, G. Sangalli, Isogeometric collocation for elastostatics and explicit dynamics, Comput. Methods Appl. Mech. Engrg. 249 (2012) 2–14.
[23] L. De Lorenzis, J.A. Evans, T.J.R. Hughes, A. Reali, Isogeometric collocation: Neumann boundary conditions and contact, Comput. Methods Appl. Mech. Engrg. 284 (2015) 21–54.
[24] D. Schillinger, J.A. Evans, A. Reali, M.A. Scott, T.J.R. Hughes, Isogeometric collocation: Cost comparison with Galerkin methods and extension to adaptive hierarchical NURBS discretizations, Comput. Methods Appl. Mech. Engrg. 267 (2013) 170–232.
[25] H. Gomez, L. De Lorenzis, The variational collocation method, Comput. Methods Appl. Mech. Engrg. 309 (2016) 152–181.
[26] M. Montardini, G. Sangalli, L. Tamellini, Optimal-order isogeometric collocation at Galerkin superconvergent points, Comput. Methods Appl. Mech. Engrg. 316 (2017) 741–757.
[27] F. Auricchio, F. Calabrò, T.J.R. Hughes, A. Reali, G. Sangalli, A simple algorithm for obtaining nearly optimal quadrature rules for NURBS-based isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 249 (2012) 15–27.
[28] M. Bartoň, V.M. Calo, Optimal quadrature rules for odd-degree spline spaces and their application to tensor-product-based isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 305 (2016) 217–240.
[29] M. Bartoň, V.M. Calo, Gauss–Galerkin quadrature rules for quadratic and cubic spline spaces and their application to isogeometric analysis, Comput. Aided Des. 82 (2017) 57–67, Isogeometric Design and Analysis.
[30] F. Calabrò, C. Manni, F. Pitolli, Computation of quadrature rules for integration with respect to refinable functions on assigned nodes, Appl. Numer. Math. 90 (2015) 168–189.
[31] F. Fahrendorf, L. De Lorenzis, H. Gomez, Reduced integration at superconvergent points in isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 328 (2018) 390–410.
[32] W. Gautschi, L. Gori, F. Pitolli, Gauss quadrature for refinable weight functions, Appl. Comput. Harmon. Anal. 8 (3) (2000) 249–257.
[33] R.R. Hiemstra, F. Calabrò, D. Schillinger, T.J.R. Hughes, Optimal and reduced quadrature rules for tensor product and hierarchically refined splines in isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 316 (2017) 966–1004.
[34] T.J.R. Hughes, A. Reali, G. Sangalli, Efficient quadrature for NURBS-based isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 199 (5) (2010) 301–313.
[35] K.A. Johannessen, Optimal quadrature for univariate and tensor product splines, Comput. Methods Appl. Mech. Engrg. 316 (2017) 84–99.
[36] D. Schillinger, S.J. Hossain, T.J.R. Hughes, Reduced Bézier element quadrature rules for quadratic and cubic splines in isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 277 (2014) 1–45.
[37] P. Antolin, A. Buffa, F. Calabrò, M. Martinelli, G. Sangalli, Efficient matrix computation for tensor-product isogeometric analysis: The use of sum factorization, Comput. Methods Appl. Mech. Engrg. 285 (2015) 817–828.

[38] A. Bressan, S. Takacs, Sum-factorization techniques in Isogeometric Analysis, arXiv preprint arXiv:1809.05471, 2018.
[39] T. Eibner, J.M. Melenk, Fast Algorithms for Setting Up the Stiffness Matrix in Hp-FEM: a Comparison, 2005.
[40] S.A. Orszag, Spectral methods for problems in complex geometrics, in: Numerical Methods for Partial Differential Equations, Elsevier, 1979, pp. 273–305.
[41] A. Mantzaflaris, B. Jüttler, Integration by interpolation and look-up for Galerkin-based isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 284 (2015) 373–400.
[42] A. Karatarakis, P. Karakitsios, M. Papadrakakis, GPU accelerated computation of the isogeometric analysis stiffness matrix, Comput. Methods Appl. Mech. Engrg. 269 (2014) 334–355.
[43] G. Sangalli, M. Tani, Matrix-free weighted quadrature for a computationally efficient isogeometric k-method, Comput. Methods Appl. Mech. Engrg. 338 (2018) 117–133.
[44] M. Ainsworth, G. Andriamaro, O. Davydov, Bernstein–Bézier finite elements of arbitrary order and optimal assembly procedures, SIAM J. Sci. Comput. 33 (6) (2011) 3087–3109.
[45] G. Karniadakis, S. Sherwin, Spectral/hp Element Methods for Computational Fluid Dynamics, Oxford University Press, 2013.
[46] P.E.J. Vos, S.J. Sherwin, R.M. Kirby, From h to p efficiently: Implementing finite and spectral/hp element methods to achieve optimal performance for low-and high-order discretisations, J. Comput. Phys. 229 (13) (2010) 5161–5181.
[47] C. De Boor, A practical guide to splines, in: Mathematics of Computation, vol. 27, Springer-Verlag, 1978.
[48] L. Schumaker, Spline Functions: Basic Theory, Cambridge University Press, 2007.
[49] M.W. Wilson, A general algorithm for nonnegative quadrature formulas, Math. Comp. 23 (106) (1969) 253–258.
[50] J. Bezanson, S. Karpinski, V.B. Shah, A. Edelman, Julia: A fast dynamic language for technical computing, arXiv preprint arXiv:1209.5145, 2012.
[51] J.R. Barber, Elasticity, Springer, 2002.
[52] S. Timoshenko, J.N. Goodier, Theory of Elasticity, McGraw-Hill book Company, 1951.
[53] A.F. Bower, Applied Mechanics of Solids, CRC press, 2009.
[54] M.A. Scott, R.N. Simpson, J.A. Evans, S. Lipton, S.P.A. Bordas, T.J.R. Hughes, T.W. Sederberg, Isogeometric boundary element analysis using unstructured T-splines, Comput. Methods Appl. Mech. Engrg. 254 (2013) 197–221.