Fast algorithms for setting up the stiffness matrix in hp-FEM: a comparison

T. Eibner J.M. Melenk

Abstract

We analyze and compare different techniques to set up the stiffness matrix in the hp-version of the finite element method. The emphasis is on methods for second order elliptic problems posed on meshes including triangular and tetrahedral elements. The polynomial degree may be variable. We present a generalization of the Spectral Galerkin Algorithm of [7], where the shape functions are adapted to the quadrature formula, to the case of triangles/tetrahedra. Additionally, we study on-the-fly matrix-vector multiplications, where merely the matrix-vector multiplication is realized without setting up the stiffness matrix. Numerical studies are included.

I. Introduction

The hp-version of the finite element method (hp-FEM) (see, e.g., the monographs [12], [11], [5] and the references therein) as well as the closely related spectral method (see, e.g., the survey article [1] and the references there) are well-established tools in computational structural and fluid mechanics. Typically, the bulk of the alphanumerical cost in the hp-FEM is in the numerical quadratures used to set up the stiffness matrix; this is in sharp contrast to the standard low order FEM (h-FEM), where most of the computational effort is spent on the solution of the resulting (linear) system. The present paper therefore considers different techniques for the rapid computation of the stiffness matrix in high order methods. As is standard in most FEM, all quadratures are done on a reference element $\widehat{K}$, which, in 2D, is either the reference square or the reference triangle; in 3D, the most important ones are the hexahedron, the tetrahedron, the prism, and the pyramid. Also the shape functions are defined on the reference element. We will discuss the generation of the element stiffness matrices for the following situation that is typical of scalar valued second order elliptic equations:

$$(S_K)_{ij} := a(\psi_j, \psi_i), \qquad a(u,v) := \int\limits_{\widehat{K}} (A(x)\nabla u) \cdot \nabla v \mathrm{d}\Omega, \tag{1}$$

where the shape functions $\psi_i \in \Psi := \{\psi_i \,|\, i = 1, \dots, N\}$ are given. The situation for vector valued second order problems (e.g., linear elasticity) is very similar as is the case of lower order terms, i.e., $\int\limits_{\widehat{K}} b(x)\nabla uv + c(x)uv\mathrm{d}\Omega$ for suitable functions b, c.

The typical procedure, especially in computational structural mechanics, is to set up the element stiffness matrices S_K of (1) for each element and then assemble them into the global stiffness matrix. This assembly procedure is described, for example, in [3], [2], [11], [5], [**?**].

If the set Ψ of shape functions spans the space of polynomials of degree p, then the simplest algorithm to compute the element stiffness matrix S has complexity $O(p^{3d})$, where d is the spatial dimension of reference element $\widehat{K}$: the number of elements in Ψ is $O(p^d)$ and $O(p^d)$ quadrature points are needed to obtain a sufficiently accurate approximation of the integrals defining the entries of S_K. It appears to be S. Orszag [10] who first pointed out that if the quadrature domain $\widehat{K}$ has product structure (i.e., if it is a square or a hexahedron) and if the shape functions Ψ have product structure, then the computational cost can be lowered to $O(p^{2d+1})$ for setting up the element stiffness matrix and $O(p^{d+1})$ to realize a fast matrix-vector multiplication. This technique is called *sum factorization*. In the 1990s, Karniadakis and Sherwin designed shape functions on triangles and tetrahedra that—after transformation to the reference square/hexahedron via the Duffy transformation—again have product structure. This insight then allowed them to extend the sum factorization idea to triangles and tetrahedra and set up the stiffness matrix in complexity $O(p^{2d+1})$ also in this case. Since modern mesh generators typically create meshes consisting of triangles/tetrahedra, this work of Karniadakis & Sherwin paved the way for the application of fast high order methods in many applications, [5].

Using sum factorization, the $O(p^{2d})$ entries of the stiffness matrix S_K are generated with work $O(p^{2d+1})$. It is natural to ask whether the optimal complexity $O(p^{2d})$ can be reached. Indeed, by adapting the shape functions to the quadature formula, it is possible to lower the complexity to $O(p^{2d})$. This is was shown for squares and hexahedra in [7]. The present paper generalizes some of these ideas to triangles and tetrahedra.

T. Eibner: Department of Mathematics, Technical University of Chemnitz, D-09107 Chemnitz, Germany, `teibner@mathematik.tu-chemnitz.de`
J.M. Melenk: Department of Mathematics, University of Reading, PO Box 220, Reading RG6 6AX, United Kingdom, `j.m.melenk@reading.ac.uk`

II. Some key ideas illustrated in 2D

A. The idea of sum factorization

The idea of sum factorization can be motivated by the following problem, which mimics the computation of a mass or a stiffness matrix in 2D: Compute, for all double indices (i_1, i_2), $(j_1, j_2) \in \{1, \ldots, p\}^2$ the field M with entries

$$M_{(i_1,i_2),(j_1,j_2)} := \sum_{k_1=0}^{q} \sum_{k_2=0}^{q} \varphi_{i_1}(k_1)\psi_{i_2}(k_2)\widetilde{\varphi}_{j_1}(k_1)\widetilde{\psi}_{j_2}(k_2)g(k_1, k_2) \tag{2}$$

where the functions φ_i, ψ_i, $\widetilde{\phi}_i$, $\widetilde{\psi}_i$, and g are given. The naive evaluation of all entries of M requires $O(p^4(1+q)^2)$ floating point operations. Since in our applications $q = O(p)$, we arrive at a total cost of $O(p^6)$ to compute the p^4 entries of M. By rerranging the sums, this work can be reduced:

$$M_{(i_1,i_2),(j_1,j_2)} = \sum_{k_1=0}^{q} \varphi_{i_1}(k_1)\widetilde{\varphi}_{j_1}(k_1)H(k_1, i_2, j_2), \qquad H(k_1, i_2, j_2) := \sum_{k_2=0}^{q} \psi_{i_2}(k_2)\widetilde{\psi}_{j_2}(k_2)g(k_1, k_2). \tag{3}$$

The cost now is $O((1+q)^2p^2)$ to set up the auxiliary field H and then $O(p^4(1+q))$ to perform the summation over k_1. Thus, the total cost is $O(p^4(1+q) + p^2(1+q)^2)$. Assuming again $q = O(p)$, we arrive at a cost $O(p^5)$. We note that the key to the lowering of the complexity from $O(p^6)$ to $O(p^5)$ is to exploit product structure. For a further lowering to the optimal complexity $O(p^4)$, additional properties must hold. In our algorithms below, we will adapt the shape functions to the quadrature formula employed. The analog of this procedure in the present example of evaluating (2) corresponds to allowing some of the sums to collapse to few terms: We define the set of relevant indices for the outer sum by

$$K_1(i_1, j_1) := \{k_1 \in \{0, \ldots, q\} \,|\, \varphi_{i_1}(k_1)\widetilde{\varphi}_{j_1}(k_1) \neq 0\}$$

and note $M_{(i_1,j_1),(i_2,j_2)} = \sum_{k_1 \in K_1(i_1,j_1)} \varphi_{i_1}(k_1)\widetilde{\varphi}_{j_1}(k_1)H(k_1, i_2, j_2)$. If we define by $|K_1| := \max\{\#K_1(i_1, j_1)| \,|\, (i_1, j_1) \in \{1, \ldots, p\}^2\}$ the maximal number of relavant terms, then the total cost to set up M is $O((1+q)^2p^2) + O(|K_1|p^4)$, which leads to the work bound $O(|K_1|p^4)$ for $q = O(p)$. If $|K_1|$ is bounded independently of p, then this represents a lowering of the total complexity of the algorithm. We observe that we arbitarily chose in (3) to evaluate the double sum defining $M_{(i_1,i_2),(j_1,j_2)}$ as the iterated sum $\sum_{k_1}\sum_{k_2}$; equally well, we could have summed $\sum_{k_2}\sum_{k_1}$. This is of interest if $|K_1|$ is not bounded independently of p but instead the set

$$K_2(i_2, j_2) := \{k_2 \in \{0, \ldots, q\} \,|\, \psi_{i_2}(k_2)\widetilde{\psi}_{j_2}(k_2) \neq 0\}$$

is small for all (i_2, j_2). If $|K_2|$ (defined analogously to $|K_1|$) is bounded independently of p, then we arrive again at an $O(p^4)$ algorithm by switching the summation order in (2) and evaluating M as

$$M_{(i_1,i_2),(j_1,j_2)} = \sum_{k_2 \in K_2(i_2,j_2)} \varphi_{i_2}(k_2)\widetilde{\varphi}_{j_2}(k_2)H(k_2, i_1, j_1), \qquad H(k_2, i_1, j_1) := \sum_{k_1=0}^{q} \psi_{i_1}(k_1)\widetilde{\psi}_{j_1}(k_1)g(k_1, k_2). \tag{4}$$

This example shows that it may be advantageous to carefully choose the summation order. It will be an ingredient of the algorithms below, and we also refer to [7] for more details.

B. Fast stiffness matrix generation in 2D

We now illustrate how the abstract ideas of sum factorization can be employed in the generation of stiffness matrices. We consider the case $\widehat{K} = \mathcal{T}^2$, where the reference triangle $\mathcal{T}^2$ is defined in Def. 3.1 below. The Duffy transformation D_2 (see Def. 3.1 below) is a bijection between the square $\mathcal{Q}^2 = (-1,1)^2$ and the triangle $\mathcal{T}^2$. The change of variables given by D_2 allows us to write the integral $a(u,v)$ in (1) is then written as

$$a(u, v) \;=\; \int_{\mathcal{Q}^2} \left(\frac{\partial_{\eta_1}\hat{u}}{1-\eta_2}, \partial_{\eta_2}\hat{u}\right) \cdot \widehat{A} \begin{pmatrix} \frac{\partial_{\eta_1}\hat{v}}{1-\eta_2} \\ \partial_{\eta_2}\hat{v} \end{pmatrix} (1-\eta_2)\, d\eta_1 d\eta_2, \qquad \hat{u} := u \circ D_2,\ \hat{v} := v \circ D_2,\ \hat{A} := A \circ D_2 \tag{5}$$

where we used the explicit formulas for D_2' and $\det D_2'$ given in Lemma 3.2. Since all quadratures will be performed on $\mathcal{Q}^2$, it will be convenient to define the shape functions Ψ only implicitly on $\mathcal{T}^2$; instead, we will define them explicitly on $\mathcal{Q}^2$ as the set Φ and then set $\Psi := \Phi \circ D_2^{-1}$. To fix ideas and notation, we define the bilinear form $\hat{a}$ and the space of polynomials $\widetilde{Q}_p$ by

$$\begin{aligned} \hat{a}(\hat{u}, \hat{v}) &:= \int_{\mathcal{Q}^2} \left(\frac{\partial_{\eta_1}\hat{u}}{1-\eta_2}, \partial_{\eta_2}\hat{u}\right) \cdot \widehat{A} \begin{pmatrix} \frac{\partial_{\eta_1}\hat{v}}{1-\eta_2} \\ \partial_{\eta_2}\hat{v} \end{pmatrix} (1-\eta_2)\, d\eta_1 d\eta_2, \\ \widetilde{Q}_p &:= \{u \in Q_p \,|\, (\partial_{\eta_1} u)|_{\eta_2=1} = 0\}, \qquad Q_p := \operatorname{span}\{\eta_1^i \eta_2^j \,|\, 0 \le i, j \le p\}. \end{aligned}$$

We note that the bilinear form $\hat{a}$ is well-defined on $\widetilde{Q}_p$ because the polynomials $\hat{u} \in \widetilde{Q}_p$ are constant on the line $\eta_2 = 1$ so that the term $\frac{1}{1-\eta_2}\partial_{\eta_1}\hat{u}$, which seemingly has a singularity at $\eta_2 = 1$, is in fact smooth there.

The basis Ψ on $\widehat{K}$ is defined such that three goals are met: a) Ψ contains the space of $\mathcal{P}_p$ of polynomial of degree p to ensure good approximation properties; b) Ψ leads to a fast evaluation of the stiffness matrix S_K; c) Ψ allows for an easy assembly of the element stiffness matrix S_K into the global stiffness matrix. The last requirement effectively dictates that the set Ψ consist of "external shape functions" (which are typically split further into "vertex shape functions" and "edge shape functions") and "internal shape functions": the internal shape functions vanish on $\partial\widehat{K}$ and the restriction to $\partial\widehat{K}$ of the external shape functions coincides with standard choices. We will construct Ψ as $\Psi = \Psi_{external} \cup \Psi_{internal}$, where, $\Psi_{external}$ is further split into $\Psi_{external} = \Psi_{vertex} \cup \Psi_{edge}$. We will assume that standard choices (e.g., those described in [11], [5]) for Ψ_{vertex} and Ψ_{edges} are made and that the number of functions in $\Psi_{external}$ is $3p$. In particular, since the transformation D_2 is polynomial and due to the property of the Duffy transformation that the edge $\eta_2 = 1$ of $\mathcal{Q}^2$ is mapped to the single point $(-1, 1)$ of $\mathcal{T}^2$, we have that the transformed external shape functions are polynomials of degree p that are constant on the edge $\eta_2 = 1$, viz.,

$$\Phi_{external} := \Psi_{external} \circ D_2 \subset \widetilde{\mathcal{Q}}_p.$$

We now seek the internal shape functions $\Psi_{internal}$ such that $\Phi_{internal} := \Psi_{internal} \circ D_2^{-1}$ is also a subset of $\widetilde{\mathcal{Q}}_p$. In view of the fact that the quadrature is done on $\mathcal{Q}^2$, we define the internal shape functions directly on $\mathcal{Q}^2$, i.e., we choose $\Phi_{internal}$ and then set $\Psi_{internal} = \Phi_{internal} \circ D_2^{-1}$. As a specific example, we take $\Phi_{internal}$ as

$$\Phi_{internal} = \{l_i^{(1)}(\eta_1) l_j^{(2)}(\eta_2) \,|\, 1 \le i, j \le p-1\},$$

where the functions $l_i^{(1)}$ and $l_j^{(2)}$ are Lagrange interpolation polynomials for point sets

$$-1 = \xi_0^{(1)} < x_1^{(1)} < \cdots < \xi_p^{(1)} = 1, \qquad -1 = \xi_0^{(2)} < \xi_1^{(2)} < \cdots < \xi_p^{(2)} = 1;$$

that is, the polynomials $l_i^{(1)}$, $l_i^{(2)}$ are given by

$$l_i^{(1)}(x) = \prod_{\substack{j=0\\ j\neq i}}^{p} \frac{x - \xi_j^{(1)}}{\xi_i^{(1)} - \xi_j^{(1)}}, \qquad l_i^{(2)}(x) = \prod_{\substack{j=0\\ j\neq i}}^{p} \frac{x - \xi_j^{(2)}}{\xi_i^{(2)} - \xi_j^{(2)}}.$$

We note that $\Phi_{internal} = \{u \in \widetilde{Q}_p \,|\, u|_{\partial\mathcal{Q}^2} = 0\}$. This in turn implies that the functions of $\Psi_{internal}$ vanish on $\partial\mathcal{T}^2$. A calculation (see, e.g., Lemma 3.8) reveals that together with standard choices of Ψ_{vertex}, Ψ_{edges} (e.g., those described in [11], [5]) we get $\Psi_{vertex} \cup \Psi_{edges} \cup \Psi_{internal} \supset \mathcal{P}_p$, which ensures that

$$\widetilde{S}_p := \text{span}\Psi \tag{6}$$

has good approximation properties.

The splitting of the shape functions Ψ into external and internal ones induces a block structure of the stiffness matrix S_K:

$$S_K = \begin{pmatrix} S^{EE} & S^{EI} \\ S^{IE} & S^{II} \end{pmatrix};$$

here the superscript E represents external shape functions and I internal ones. We now illustrate how this choice of internal shape functions allows us to construct the stiffness matrix S_K in optimal complexity by showing how S^{II} can be computed with work $O(p^4)$; analogous calculations can be done for the other blocks.

We define a tensor product quadrature formula $\mathfrak{S}_q$ by

$$\mathfrak{S}_q(g) := \sum_{k=0}^{q}\sum_{l=0}^{q} \omega_k^{(1)}\omega_l^{(2)} g(x_k^{(1)}, x_l^{(2)}) \approx \int_{\mathcal{Q}^2} g(\eta_1, \eta_2)(1-\eta_2)\,d\eta_1 d\eta_2$$

and replace the integral in the definition of $\hat{a}$ by this quadrature formula. Then, the entries of the matris S^{II} are given by (we use double indices (i_1, i_2), (j_1, j_2) for notational convenience)

$$S^{II}_{(i_1,i_2),(j_1,j_2)} = \sum_{k=0}^{q}\sum_{l=0}^{q} \omega_k^{(1)}\omega_l^{(2)} \left(l'_{i_1}(x_k^{(1)}) \frac{l_{i_2}(x_l^{(2)})}{1 - x_l^{(2)}} ,\, l_{i_1}(x_k^{(1)}) l'_{i_2}(x_l^{(2)}) \right) \hat{A}(x_k^{(1)}, x_l^{(2)}) \begin{pmatrix} l'_{j_1}(x_k^{(1)}) \frac{l_{j_2}(x_l^{(2)})}{1-x_l^{(2)}} \\ l_{j_1}(x_k^{(1))} l'_{j_2}(x_l^{(2)}) \end{pmatrix}.$$

Remark 2.1: Below, we will be interested in the case that the 1D quadrature rules $\sum_{k=0}^{q} \omega_k^{(1)} f(x_k^{(1)})$, $\sum_{l=0}^{q} \omega_l^{(2)} f(x_l^{(2)})$ are of Gauss-Lobatto and Gauss-Lobatto-Jacobi type. For these quadrature rules, the endpoints ± 1 are quadrature knots. Evaluating terms of the form $\frac{l_{i_2}^{(2)}(x)}{1-x}$ at the $x = 1$ is then to be understood as taking the limit as $x \to 0$. This is merely a notational problem since the polynomials $l_{i_2}^{(2)}$, $l_{j_2}^{(2)}$ vanish at the endpoints $x \pm 1$.

Upon expanding the matrix vector products inside the double sum we arrive at a sum with 4 four terms. For the technique of sum factorization, all four terms can be treated similarly. For example, for one of the "mixed" ones, we get

$$S^{II,mixed}_{(i_1,i_2),(j_1,j_2)} := \sum_{k=0}^{q}\sum_{l=0}^{q} \omega_k^{(1)}\omega_l^{(2)} l'_{i_1}(x_k^{(1)}) \frac{l_{i_2}(x_l^{(2)})}{1-x_l^{(2)}} \hat{A}_{12}(x_k^{(1)},x_l^{(2)}) l_{j_1}(x_k^{(1)}) \frac{l'_{j_2}(x_l^{(2)})}{1-x_l^{(2)}}.$$

Using sum factorization techniques, the cost to evaluate all $(p-1)^4$ entries of $S^{II,mixed}$ is, as seen above, $O(p^4(1+q)+p^2(1+q)^2)$. Up to now, the fact that the functions $l_i^{(1)}$, $l_i^{(2)}$ are Lagrange interpolation polynomials with respect to some points was not relevant. If we choose the points $\{\xi_i^{(1)}\,|\,i=0,\ldots,p\}$, $\{\xi_i^{(2)}\,|\,i=0,\ldots,p\}$ to be subsets of the quadrature points, then the sets

$$\{k \in \{0,\ldots,q\}\,|\, l_{i_1}(x_k^{(1)}) \neq 0\}, \qquad \{l \in \{0,\ldots,q\}\,|\, l_{i_2}(x_l^{(2)}) \neq 0\},$$

have cardinality bounded by $1+(q-p)$. Thus, from the above discussion, we see that the cost to set up the $S^{II,mixed}$ reduced to $O(p^2q^2+p^4(1+q-p))$. In particular, if $q=p+m$ for a fixed m, we arrive at the desired optimal complexity $O(p^4)$. One way to proceed therefore is to choose a quadrature formula with $q=p+m$ and then to select from the quadrature knots an appropriate subset for the definition of the Lagrange interpolation polynomials $l_i^{(1)}$, $l_i^{(2)}$. A special case is that of $q=p$: then the choice of the quadrature formula dictates uniquely the polynomials $l_i^{(1)}$, $l_i^{(2)}$.

Remark 2.2: If $q>p$, then the interpolation points $\xi_i^{(1)}$, $\xi_i^{(2)}$ are not uniquely determined and a selection has to be made. One possible criterion is the conditioning of the resulting mass matrix; we refer to [7] where similar considerations were performed for selecting points on squares and hexahedra.

C. Convergence properties

1) Gauss-Lobatto and Gauss-Lobatto-Jacobi quadrature: The use of quadrature formulas entails errors that need to be estimated. We will consider quadrature formulas that are tensor products of Gauss-Lobatto and Gauss-Lobatto-Jacobi quadrature rules. We note that this choice implies in particular that the endpoints $x=\pm 1$ are in fact quadrature points. More specifically, we take $x_i^{(1)} = x_i^{(GL)}$, $i=0,\ldots,q$, and $x_i^{(2)} = x_i^{(GLJ)}$, $i=0,\ldots,q$, where the Gauss-Lobatto points $-1 = x_0^{GL} < x_1^{GL} < \cdots < x_q^{GL} = 1$ are the zeros of the polynomial $x \mapsto (1-x^2)P_{q-1}^{(1,1)}(x)$ and the Gauss-Lobatto-Jacobi points $-1 = x_0^{GLJ} < x_1^{GLJ} < \cdots < x_q^{GLJ} = 1$ are the zeros of $x \mapsto (1-x^2)P_{q-1}^{(2,1)}(x)$; here, we employed the standard notation for the Jacobi polynomials $P_q^{(\alpha,\beta)}$. It is possible to find positive quadrature weights ω_i^{GL}, ω_i^{GLJ} (see, e.g., [5, Appendix B] for explicit formulas) such that

$$\sum_{i=0}^{q}\omega_i^{GL} f(x_i^{GL}) = \int_{-1}^{1} f(x)\,dx, \qquad \sum_{i=0}^{q}\omega_i^{GLJ} f(x_i^{GLJ}) = \int_{-1}^{1} f(x)(1-x)\,dx \qquad \forall f \in \mathcal{P}_{2q-1}, \tag{7a}$$

$$\frac{1}{3}\sum_{i=0}^{q}\omega_i^{GL} |f(x_i^{GL})|^2 \le \int_{-1}^{1} |f(x)|^2\,dx \le \sum_{i=0}^{q}\omega_i^{GL}|f(x_i^{GLJ})|^2 \qquad \forall f \in \mathcal{P}_q, \tag{7b}$$

$$\frac{1}{4}\sum_{i=0}^{q}\omega_i^{GLJ} |f(x_i^{GLJ})|^2 \le \int_{-1}^{1} |f(x)|^2(1-x)\,dx \le \sum_{i=0}^{q}\omega_i^{GLJ}|f(x_i^{GLJ})|^2 \qquad \forall f \in \mathcal{P}_q. \tag{7c}$$

We may then define the quadrature rule

$$\widehat{\mathfrak{S}}_q(g) := \sum_{k=0}^{q}\sum_{l=0}^{q}\omega_k^{GL}\omega_l^{GLJ} g(x_k^{GL}, x_l^{GLJ}) \approx \int_{\mathcal{Q}^2} g(\eta_1,\eta_2)(1-\eta_2)\,d\eta_1 d\eta_2,$$

on the square $\mathcal{Q}^2$, which in turn defines a quadrature formula on $\mathcal{T}^2$ via

$$\mathfrak{S}_q(g) := \widehat{\mathfrak{S}}_q(g \circ D_2) \approx \int_{\mathcal{T}^2} g(x,y)\,dx\,dy.$$

The bilinear form $a(\cdot,\cdot)$ of (1) may then be replaced with its discrete counterpart

$$a_q(u,v) := \mathfrak{S}_q(\nabla u \cdot A \nabla v).$$

The properties (7) of the Gauss-Lobatto and the Gauss-Lobatto-Jacobi quadrature allow us to formulate a coercivity result for a_q:

Theorem 2.3: Let $\widehat{K} = \mathcal{T}^2$, let $A \in L^\infty(\widehat{K})$ be a matrix-valued function $x \mapsto A(x) \in \mathbb{R}^{2\times 2}$ such that for each $x \in \mathcal{T}^2$ the matrix $A(x)$ is symmetric positive definite with $0 < \lambda \le A(x) \le \Lambda$. Then

$$\frac{\lambda}{12}\|\nabla u\|^2_{L^2(\mathcal{T}^2)} \le a_q(u,u) \le \Lambda \|\nabla u\|^2_{L^2(\mathcal{T}^2)} \qquad \forall u \text{ such that } u \circ D_2 \in \widetilde{Q}_q.$$

We note that the shape functions Ψ defined above span a $\widetilde{S}_p$ of the form considered in Theorem 2.3. We also note that the case $q=p$ is explicitly included.

2) Convergence analysis: The discrete coercivity result of Theorem 2.3 allows one to perform a quadrature analysis based on the Strang lemma—we refer to, e.g., [6], [1], [8] where the details are elaborated. In order to illustrate the kind of result that can be expected, we formulate how the optimal algebraic rate of convergence of the p-version FEM is preserved. To that end, we consider the model problem

$$\nabla \cdot (A(x)\nabla u) = f \quad \text{ on } \Omega, \qquad u|_{\partial\Omega} = 0 \tag{8}$$

and let $\mathcal{T}$ be a triangulation (with element maps $F_K : \mathcal{T}^2 \to K$) of a domain $\Omega \subset \mathbb{R}^2$; we recall the definition of $\widetilde{S}_p$ in (6). The approximation space V_p is obtained by assembling the shape functions Ψ in the standard way, i.e., $V_p = \{u \in H^1_0(\Omega) \,|\, u|_K \circ F_K \in \widetilde{S}_p \quad \forall K \in \mathcal{T}\}$. The fully discrete scheme consists of the finite element method for (8) with approximation space V_p where all integrals are replaced with the quadrature formula $\mathfrak{S}_q$: Find $u_{p,q} \in V_p$ such that

$$\sum_{K\in\mathcal{T}} \mathfrak{S}_q((\nabla u_{p,q} A \nabla v) \circ F_K) = \sum_{K\in\mathcal{T}} \mathfrak{S}_q((fv)\circ F_K) \qquad \forall v \in V_p. \tag{9}$$

The following result illustrates that the presence of quadrature does not affect the rate of convergence of the p-version:

Theorem 2.4: Let $\Omega \subset \mathbb{R}^2$ be a domain with piecewise analytic boundary. Let $\mathcal{T}$ be a (fixed) triangulation with analytic element maps. Let the matrix A of (1) be analytic on $\overline{\Omega}$ and assume that $0 < \lambda \le A(x) \le \Lambda < \infty$ for all $x \in \Omega$. Let f be analytic on $\overline{\Omega}$. Let $u \in H^1_0(\Omega)$ be the solution of (8) and $u_{p,q}$ be the solution of (9). Then for $q \ge p$ the approximation $u_{p,q}$ exists and satisfies

$$\|u - u_{p,q}\|_{H^1(\Omega)} \le C \left\{ \inf_{v\in V_{p/2}} \|u - v\|_{H^1(\Omega)} + e^{-bp} \right\},$$

where the constants C, $b > 0$ depend only on λ, Λ, Ω, the elements maps F_K, and f.

Remark 2.5: The solution u of the elliptic boundary value problem in Corollary 2.4 has singularities at the vertices of the curvilinear polygon Ω. This implies that the optimal rate of convergene of the p-FEM (without quadrature errors) is algebraic. Corollary 2.4 states that the fully discrete scheme converges at athe same (algebraic) rate. If the solution happens to be analytic, then the fully discrete scheme will also convergence at an exponential rate.

D. Overintegration

We showed in Theorem 2.3 that even the minimial quadrature rule (i.e., the case $q = p$) discussed above retains for *scalar problems* certain coercivity properties of the continuous problem. However, the proof of Theorem 2.3 suggests that the coercivity constant of the fully discrete scheme deteriorates in the presence of distorted meshes. Additionally, in the case of non-affine meshes a corresponding analysis for vector-valued problems such as the system of linear elasticity is, to the knowledge of the authors, missing. These are just some reason why it is customary in computational structural mechanics to employ *overintegration*, where the number of quadrature points q is strictly greater than the polynomial degree p. Typically, $q \in \{p+1, \dots, p+4\}$. In this situation, it was proposed in [7] to fix a quadrature order q and then select a subset of the quadrature points as the knots on which to base the definition of the Lagrange interpolation polynomials. Some criteria on on which to base the selection of the points are discussed in [7] and could be extended to the present situation. However, the focus of the present paper is not the optimal choice of the points. Instead, we concentrate on investigating whether adapting the shape functions to the quadrature formula (and thereby reducing the complexity of setting up the stiffness matrix) can compete with the use of standard shape functions such as those discussed in [5]: We note that the number of internal shape functions proposed here is roughly twice that of the standard choice for 2D problems and roughly six times that of standard choices in 3D. Since in many hp-FEM implementations the internal shape functions are eliminated on the element level by Gaussian elimination with work $O(p^{3d})$, the savings achieved by fast quadrature using enlarged sets of internal shape functions may be partially offset by a cost increase in the static condensation.

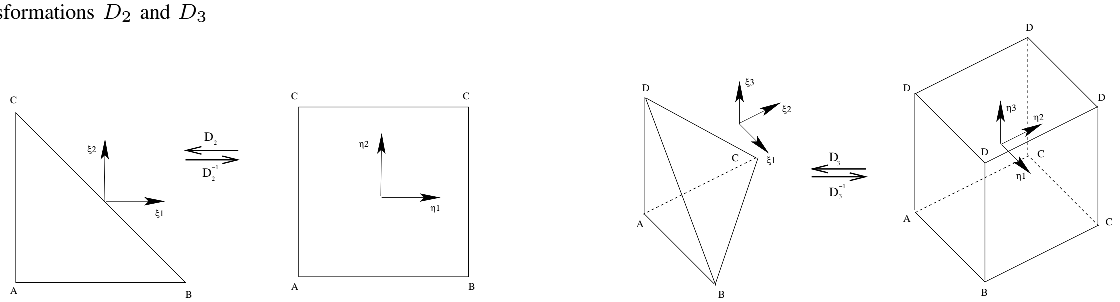
We also mention that the shape functions that we study below differ slightly from those discussed so far. Our reasons for the specific choice made below is that it ensures that the second derivatives of the shape functions (on $\mathcal{T}^d$) are sufficiently smooth. This is convenient, for example, if the element residual has to be computed in residual based error estimation as proposed in [9].

III. Shape functions on triangles and tetrahedra

A. Duffy transformation

The coefficient matrix A in (1) is often non-polynomial due to, for example, the use of blending elements to capture non-polynomial geometries. In this situation, the entries of S_K cannot be evaluated exactly and numerical quadrature has to be employed. If the reference element $\widehat{K}$ is the reference triangle or tetrahedron, then it is natural to perform the quadrature on a square/hexahedron by a further transformation using the Duffy transformation. The following definition formalizes these notions:

Fig. 1. Transformations D_2 and D_3



Definition 3.1 (reference elements): We define the triangle, tetrahedron, and the i-th dimensional reference cube $\mathcal{Q}^i$ by:

$$\begin{aligned}
\mathcal{T}^2 &= \{(x,y) \mid -1 < x,y \ \wedge \ x+y < 0\}, \\
\mathcal{T}^3 &= \{(x,y,z) \mid -1 < x,y,z \ \wedge \ x+y+z < -1\}, \\
\mathcal{Q}^i &= (-1,1)^i.
\end{aligned}$$

Lemma 3.2: The Duffy transformations D_2, D_3 are given by:

$$D_2 \ : \ (\eta_1,\eta_2) \mapsto \left(\frac{1}{2}(1+\eta_1)(1-\eta_2)-1,\eta_2\right), \tag{10}$$

$$D_3 \ : \ (\eta_1,\eta_2,\eta_3) \mapsto \left(\frac{1}{4}(1+\eta_1)(1-\eta_2)(1-\eta_3)-1, \frac{1}{2}(1+\eta_2)(1-\eta_3)-1,\eta_3\right). \tag{11}$$

Then

$$|\det D_2'| = \left(\frac{1-\eta_2}{2}\right), \qquad |\det D_3'| = \left(\frac{1-\eta_2}{2}\right)\left(\frac{1-\eta_3}{2}\right)^2, \qquad \textit{and} \quad \mathcal{T}^i = D_i(\mathcal{Q}^i), \quad i=2,3.$$

If $\widehat{K} = \mathcal{T}^d$, then the chain rule allows us to rewrite the integral (1) as follows:

$$a(u,v) = \int\limits_{\mathcal{T}^d} (A(x)\nabla u)\cdot\nabla v \mathrm{d}\Omega = \int\limits_{\mathcal{Q}^d} \Big((\nabla u \circ D_d), \hat{A}(\nabla v \circ D_d)\Big)\, |\det D_d'| \mathrm{d}\Omega, \qquad \hat{A} := (D_d')^{-1}(A\circ D_d)(D_d')^{-T}. \tag{12}$$

Once the integration over $\mathcal{T}^d$ is rewritten as an integration over $\mathcal{Q}^d$, we may employ standard tensor product quadrature techniques of Gauss, Gauss-Lobatto, or, more generally, of Gauss-Jacobi-Lobatto type.

B. Shape functions on $\mathcal{T}^2$ and $\mathcal{T}^3$

The stiffness matrix S_K is completely fixed once the shape functions Ψ are chosen. We will discuss two different choices of this set Ψ below. Some key considerations that determine the construction are:

- Since the quadrature over $\mathcal{T}^d$ is reformulated as a quadrature over $\mathcal{Q}^d$ in (12), it is advantageous to define the shape functions explicitly as functions on $\mathcal{Q}^d$ and thereby only implicitly on $\mathcal{T}^d$ by means of the Duffy transformation. In view of the fact that sum factorization techniques will employed, the shape functions (as defined on $\mathcal{Q}^d$) have product structure.
- In order to facilitate variable polynomial degree distributions in hp-FEM, a (possibly different) polynomial degree is associated with each of the topological entities, i.e., each edge e, face f (in 3D), and the element has its own degree.

We will discuss two sets of shape functions. The first set $\Psi^{(KS)}$ is taken from [5] and contains shape functions with a tensor product structure suitable for applying sum factorization. The second set $\Psi^{(Lag)}$ is a modification of $\Psi^{(KS)}$ that contains shape functions where the so-called internal shape functions are adapted to the quadrature rules. First some abbreviations:
Abbreviations 3.3: Let

$$f_1(x) := \left(\frac{1-x}{2}\right)\left(\frac{1+x}{2}\right), \quad f_2(x) := \left(\frac{1-x}{2}\right), \quad f_3(x) := \left(\frac{1+x}{2}\right).$$

Definition 3.4 (shape functions on $\mathcal{T}^2$): Let $\mathcal{T}^2$ be the reference triangle with vertices A, B, C. Let $\mathbf{p} = (p_{AB}, p_{AC}, p_{BC}, p_K)$ be a degree vector with the understanding that p_{AB} is the polynomial degree associated with the edge AB etc. The value p_K is the degree associated with the element. For $i=1,2$ let $N_i = \{\eta_k^{(i)} | k = 1,..,p_K - i\}$ be a nodal set with $-1 < \eta_1^{(i)} < \ldots < \eta_{P_K-i}^{(i)} < 1$ and $l_k^{(N_i)}$ the k-th Lagrange interpolation polynomial with respect to N_i. Then we define

$$\Psi^{(KS)} = \bigcup_{B=0}^{5} \Psi_B^{(KS)} \quad \text{and} \quad \Psi^{(Lag)} = \bigcup_{B=0}^{5} \Psi_B^{(Lag)},$$

where

$$\Psi_B^{(KS)} := \Phi_B^{(KS)} \circ D_2^{-1} = \left\{\phi \circ D_2^{-1} | \phi \in \Phi_B^{(KS)}\right\}, \qquad \Psi_B^{(Lag)} := \Phi_B^{(Lag)} \circ D_2^{-1} = \left\{\phi \circ D_2^{-1} | \phi \in \Phi_B^{(Lag)}\right\}$$

and the sets Φ_B are sets of functions defined on $\mathcal{Q}^2$ given by

$$\begin{aligned}
\Phi_0^{(KS)} = \Phi_0^{(Lag)} = \Phi_0 &:= \{f_3(\eta_2)\}, \\
\Phi_1^{(KS)} = \Phi_1^{(Lag)} = \Phi_1 &:= \{f_2(\eta_1)f_2(\eta_2), f_3(\eta_1)f_2(\eta_2)\}, \\
\Phi_2^{(KS)} &:= \left\{f_1(\eta_1)f_2^{p+1}(\eta_2)P_{p-1}^{(1,1)}(\eta_1) \mid p = 1, \dots, p_{AB} - 1\right\}, \\
\Phi_2^{(Lag)} &:= \left\{f_1(\eta_1)f_2^{2}(\eta_2)P_{p-1}^{(1,1)}(\eta_1) \mid p = 1, \dots, p_{AB} - 1\right\}, \\
\Phi_3^{(KS)} = \Phi_3^{(Lag)} = \Phi_3 &:= \left\{f_2(\eta_1)f_1(\eta_2)P_{q-1}^{(1,1)}(\eta_2) \mid q = 1, \dots, p_{AC} - 1\right\}, \\
\Phi_4^{(KS)} = \Phi_4^{(Lag)} = \Phi_4 &:= \left\{f_3(\eta_1)f_1(\eta_2)P_{q-1}^{(1,1)}(\eta_2) \mid q = 1, \dots, p_{BC} - 1\right\}, \\
\Phi_5^{(KS)} &:= \left\{f_1(\eta_1)f_1(\eta_2)f_2^{p}(\eta_2)P_{p-1}^{(1,1)}(\eta_1)P_{q-1}^{(2p+1,1)}(\eta_2) \middle| \begin{array}{l} 1 \le p \le p_K - 2 \\ 1 \le q \le p_K - p - 1 \end{array}\right\}, \\
\Phi_5^{(Lag)} &:= \left\{f_1(\eta_1)f_1(\eta_2)f_2(\eta_2)C_p l_p^{(N_1)}(\eta_1)C_q l_q^{(N_2)}(\eta_2) \middle| \begin{array}{l} 1 \le p \le p_K - 1 \\ 1 \le q \le p_K - 2 \end{array}\right\},
\end{aligned}$$

where the constants C_p are scaling parameters.

Definition 3.5 (shape functions on $\mathcal{T}^3$): Let $\mathcal{T}^3$ be the reference tetrahedron and let $\mathbf{p} = (p_{AB}, \dots, p_{CD}, p_{ABC}, \dots, p_{BCD}, p_K)$ be a degree vector. As in the 2D case, the subscripts $AB, \dots, BCD$ represent edges and faces of the tetrahedron $\mathcal{T}^3$ with vertices A, B, C, D. For $i = 1, 2, 3$ let $N_i = \{\eta_k^{(i)} | k = 1, .., p_K - 3\}$ be a nodal set with and $l_k^{(N_i)}$ the k-th Lagrange interpolation polynomial with respect to N_i. Then we define

$$\Psi^{(KS)} = \bigcup_{B=0}^{13} \Psi_B^{(KS)} \quad \text{and} \quad \Psi^{(Lag)} = \bigcup_{B=0}^{13} \Psi_B^{(Lag)},$$

where

$$\Psi_B^{(KS)} := \Phi_B^{(KS)} \circ D_3^{-1} = \left\{\phi \circ D_2^{-1} | \phi \in \Phi_B^{(KS)}\right\} \qquad \Psi_B^{(Lag)} := \Phi_B^{(Lag)} \circ D_3^{-1} = \left\{\phi \circ D_2^{-1} | \phi \in \Phi_B^{(Lag)}\right\}$$

and the sets Φ_B are sets of functions defined on $\mathcal{Q}^3$ given by:

$$\begin{aligned}
\Phi_0^{(KS)} = \Phi_0^{(Lag)} = \Phi_0 &:= \{f_3(\eta_3)\}, \\
\Phi_1^{(KS)} = \Phi_1^{(Lag)} = \Phi_1 &:= \{f_3(\eta_2)f_2(\eta_3)\}, \\
\Phi_2^{(KS)} = \Phi_2^{(Lag)} = \Phi_2 &:= \{f_2(\eta_1)f_2(\eta_2)f_2(\eta_3), f_3(\eta_1)f_2(\eta_2)f_2(\eta_3)\}, \\
\Phi_3^{(KS)} = \Phi_3^{(Lag)} = \Phi_3 &:= \left\{f_1(\eta_1)P_{p-1}^{(1,1)}(\eta_1)f_2^{p+1}(\eta_2)f_2^{p+1}(\eta_3) \mid 1 \le p \le p_{AB} - 1\right\}, \\
\Phi_4^{(KS)} = \Phi_4^{(Lag)} = \Phi_4 &:= \left\{f_2(\eta_1)f_1(\eta_2)P_{q-1}^{(1,1)}(\eta_2)f_2^{q+1}(\eta_3) \mid 1 \le q \le p_{AC} - 1\right\}, \\
\Phi_5^{(KS)} = \Phi_5^{(Lag)} = \Phi_5 &:= \left\{f_3(\eta_1)f_1(\eta_2)P_{q-1}^{(1,1)}(\eta_2)f_2^{q+1}(\eta_3) \mid 1 \le q \le p_{BC} - 1\right\}, \\
\Phi_6^{(KS)} = \Phi_6^{(Lag)} = \Phi_6 &:= \left\{f_2(\eta_1)f_2(\eta_2)f_1(\eta_3)P_{r-1}^{(1,1)}(\eta_3) \mid 1 \le r \le p_{AD} - 1\right\}, \\
\Phi_7^{(KS)} = \Phi_7^{(Lag)} = \Phi_7 &:= \left\{f_3(\eta_1)f_2(\eta_2)f_1(\eta_3)P_{r-1}^{(1,1)}(\eta_3) \mid 1 \le r \le p_{BD} - 1\right\}, \\
\Phi_8^{(KS)} = \Phi_8^{(Lag)} = \Phi_8 &:= \left\{f_3(\eta_2)f_1(\eta_3)P_{r-1}^{(1,1)}(\eta_3) \mid 1 \le r \le p_{CD} - 1\right\}, \\
\Phi_9^{(KS)} = \Phi_9^{(Lag)} = \Phi_9 &:= \Big\{f_1(\eta_1)P_{p-1}^{(1,1)}(\eta_1)f_2^{p}(\eta_2)f_1(\eta_2)P_{q-1}^{(2p+1,1)}(\eta_2)f_2^{p+q+1}(\eta_3)\Big| \\
&\qquad\qquad 1 \le p \le p_{ABC} - 2,\ 1 \le q \le p_{ABC} - p - 1\Big\}, \\
\Phi_{10}^{(KS)} = \Phi_{10}^{(Lag)} = \Phi_{10} &:= \Big\{f_1(\eta_1)P_{p-1}^{(1,1)}(\eta_1)f_2^{p+1}(\eta_2)f_1(\eta_3)f_2^{p}(\eta_3)P_{r-1}^{(2p+1,1)}(\eta_3)\Big| \\
&\qquad\qquad 1 \le p \le p_{ABD} - 2,\ 1 \le r \le p_{ABD} - p - 1\Big\}, \\
\Phi_{11}^{(KS)} = \Phi_{11}^{(Lag)} = \Phi_{11} &:= \Big\{f_2(\eta_1)f_1(\eta_2)P_{q-1}^{(1,1)}(\eta_2)f_1(\eta_3)f_2^{q}(\eta_3)P_{r-1}^{(2q+1,1)}(\eta_3)\Big| \\
&\qquad\qquad 1 \le q \le p_{ACD} - 2,\ 1 \le r \le p_{ACD} - q - 1\Big\},
\end{aligned}$$

$$
\begin{aligned}
\Phi_{12}^{(KS)} = \Phi_{12}^{(Lag)} = \Phi_{12} &:= \Big\{ f_3(\eta_1) f_1(\eta_2) P_{q-1}^{(1,1)}(\eta_2) f_1(\eta_3) f_2^q(\eta_3) P_{r-1}^{(2q+1,1)}(\eta_3) \Big| \\
&\qquad 1 \le q \le p_{BCD} - 2,\ 1 \le r \le p_{BCD} - q - 1 \Big\}, \\
\Phi_{13}^{(KS)} &:= \Big\{ f_1(\eta_1) P_{p-1}^{(1,1)}(\eta_1) f_1(\eta_2) f_2^p(\eta_2) P_{q-1}^{(2p+1,1)}(\eta_2) f_1(\eta_3) f_2^{p+q}(\eta_2) P_{r-1}^{(2p+2q+1,1)}(\eta_3) \Big| \\
&\qquad 1 \le p \le p_K - 3,\ 1 \le q \le p_K - p - 2,\ 1 \le r \le p_K - p - q - 1 \Big\}, \\
\Phi_{13}^{(Lag)} &:= \Big\{ f_1(\eta_1) C_p l_p^{(1)}(\eta_1) f_1(\eta_2) f_2(\eta_2) C_q l_q^{(2)}(\eta_2) f_1(\eta_3) f_2^2(\eta_3) C_r l_r^{(3)}(\eta_3) \Big| \\
&\qquad 1 \le p \le p_K - 3,\ 1 \le q \le p_K - 3,\ 1 \le r \le p_K - 3 \Big\}.
\end{aligned}
$$

Here, the factor C_p is a scaling factor. We recall that the sets Φ_i define the shape functions on $\mathcal{Q}^d$;

Remark 3.6: Restricted to the boundary $\partial\mathcal{T}^d$ the shape functions of $\Phi^{(Lag)}$ and $\Phi^{(KS)}$ are identical. The major difference between these two sets lies in the internal shape functions and the number of internal shape functions:

$$
\#\mathrm{INT}(\Phi^{(Lag)}) = \begin{cases} (p_K-1)(p_K-2) & : \ d=2 \\ (p_K-3)^3 & : \ d=3 \end{cases}, \quad \#\mathrm{INT}(\Phi^{(KS)}) = \begin{cases} \frac{1}{2}(p_K-1)(p_K-2) & : \ d=2 \\ \frac{1}{6}(p_K-1)(p_K-2)(p_K-3) & : \ d=3 \end{cases}.
$$

Remark 3.7: In the 2-dimensional case we additionally modified the shape functions corresponding to the edge AB. However, restricted to $\partial\mathcal{T}^2$ the functions of $\Phi_2^{(Lag)}$ and $\Phi_2^{(KS)}$ are still identical but due to this modification the index p becomes relevant exclusively for the η_1-component which in turn allows a very effective implementation of the spectral Galerkin method below. Since for the 3-dimensional case such a simplification of the structur cannot be obtained without changing the shape functions on $\partial\mathcal{T}^3$ significantly we restrict our attention here to modifying the internal shape functions.

The subdivision of the shape functions into different groups in Definition 3.4 and Definition 3.5 follows a standard pattern in hp-FEM. In the case of the triangle we have the vertex shape functions $\psi \in \Psi_0 \cup \Psi_1$, the edge shape functions $\psi \in \Psi_2 \cup \ldots \cup \Psi_4$ and the internal shape functions $\psi \in \Psi_5$. For the tetrahedron we have the vertex shape functions $\psi \in \Psi_0 \cup \ldots \cup \Psi_2$, the edge shape functions $\psi \in \Psi_2 \cup \ldots \cup \Psi_8$, the face shape functions $\psi \in \Psi_9 \cup \ldots \cup \Psi_{12}$ and the internal shape functions $\psi \in \Psi_{13}$. The following lemma collects the important properties of the shape functions:

Lemma 3.8: For $d = 2,3$ let $p(K)$ be the polynomial degree distribution of $K \in \mathcal{T}$. Let $\Psi^{(KS)} = \Phi^{(KS)} \circ D_d^{-1}$ and $\Psi^{(Lag)} = \Phi^{(Lag)} \circ D_d^{-1}$ be given by Definition 3.4 or Definition 3.5. Denote by

$$
\mathcal{P}_p := \operatorname{span} \left\{ \prod_{i=1}^{d} x_i^{\alpha_i} \mid \alpha_i \in \mathbb{N}_0, \quad \sum_{i=1}^{d} \alpha_i \le p \right\}
$$

the spaces of all polynomials of total degree p in d variables and set

$$
\mathcal{P}_{\mathbf{p}}(\mathcal{T}^d) = \begin{cases} \{\ \psi \in \mathcal{P}_{p_K}(\mathcal{T}^d) \mid \psi|_e \in \mathcal{P}_{p_e}(e)\ \forall e = \text{edge of } \mathcal{T}^2 \} & : \ d=2 \\ \left\{ \psi \in \mathcal{P}_{p_K}(\mathcal{T}^d) \ \middle| \ \begin{array}{l} \psi|_e \in \mathcal{P}_{p_e}(e)\ \forall e = \text{edge of } \mathcal{T}^3 \\ \psi|_f \in \mathcal{P}_{p_f}(f)\ \forall f = \text{face of } \mathcal{T}^3 \end{array} \right\} & : \ d=3 \end{cases}.
$$

Then

1) *The sets $\Psi^{(KS)}$ and $\Psi^{(Lag)}$ are sets of linearly independent functions.*
2) *$\mathcal{P}_{\mathbf{p}}(\mathcal{T}^d) = \operatorname{span}\{\psi \mid \psi \in \Psi^{(KS)}\} \subset \operatorname{span}\{\psi \mid \psi \in \Psi^{(Lag)}\} =: \tilde{\mathcal{Q}}_{\mathbf{p}}(\mathcal{T}^d)$.*
3) *All $\psi \in \Psi^{(KS)}$ are polynomial.*
4) *For arbitrary $\psi = \psi(\xi) \in \Psi^{(Lag)}$ and $D_d : \mathbb{R}^d \to \mathbb{R}^d : \eta \mapsto \xi$ given by Lemma 3.2*

$$
\left[\frac{\partial \psi}{\partial \xi_i} \right] \circ D_d \quad \text{and} \quad \left[\frac{\partial^2 \psi}{\partial \xi_i \partial \xi_j} \right] \circ D_d
$$

are polynomial for all $i, j \in \{1, .., d\}$.

Proof: The properties concerning $\Psi^{(KS)}$ are shown in [5]. The properties of $\Psi^{(Lag)}$ follow by some calculations, the details of which can be found in [4]. ■

Corollary 3.9: For $d = 2,3$ let $\Psi^{(KS/Lag)}$ and $\Phi^{(KS/Lag)}$ be given by Definition 3.4 or Definition 3.5 respectively. Then the entries of the local stiffness matrix S_K, given by (12), can be computed as

$$
(S_K)_{ij} \;=\; \int_{\mathcal{Q}^d} \left(\tilde{\nabla}\phi_j, \hat{C}\tilde{\nabla}\phi_i \right) |\det D_d'| d\Omega = \sum_{r,r'=1}^{3} \int_{\mathcal{Q}^d} \tilde{\nabla}_{r'}\phi_j \hat{C}_{r'r} \tilde{\nabla}_r \phi_i |\det D_d'| d\Omega
$$

where

$$
\tilde{\nabla}\phi_i^{(K)} := \begin{cases} \left[\frac{1}{(1-\eta_2)} \frac{\partial \phi}{\partial \eta_1}, \frac{\partial \phi}{\partial \eta_2} \right]^T & : \ d=2 \\ \left[\frac{1}{(1-\eta_2)(1-\eta_3)} \frac{\partial \phi}{\partial \eta_1}, \frac{1}{(1-\eta_3)} \frac{\partial \phi}{\partial \eta_2}, \frac{\partial \phi}{\partial \eta_3} \right]^T & : \ d=3 \end{cases}
$$

is polynomial and

$$\hat{C} := M_d^{-1}(A \circ D_d) M_d^{-T}, \qquad M_2^{-1} := \begin{bmatrix} 2 & 2(1+\eta_1) \\ 0 & 1 \end{bmatrix}, \qquad M_3^{-1} := \begin{bmatrix} 4 & 2(1+\eta_1) & 2(1+\eta_1) \\ 0 & 2 & (1+\eta_2) \\ 0 & 0 & 1 \end{bmatrix}.$$

IV. Approximation properties

Lemma 3.8 implies that the space spanned by the Karniadakis-Sherwin shape functions is contained in the space spanned by the Lagrange shape functions. For problems where the FEM realizes an energy minimization this implies that the FEM based on the Lagrange shape functions will have improved approximation properties. The following example illustrates this effect.
Example 4.1: For $\Omega = \mathcal{T}^d$ let

$$-\Delta u = 1 \quad \text{on } \Omega \quad \text{and} \quad u = 0 \quad \text{on } \partial\Omega.$$

We apply the p-version FEM on a single element using both the sets $\Phi^{(KS)}$ and $\Phi^{(Lag)}$. The results of our computations are shown in Figure 5. As expected, using $\Phi^{(Lag)}$ reduces the error significantly. The rate of convergence, however, cannot be expected to be improved since both sets of shape functions rely on the approximation properties of polynomials.

V. Algorithms for setting up the element stiffness matrices

Now, after we defined several sets of shape functions, we will consider different algorithms for setting up the element stiffness matrix. We start with the most elementary one, which applies a standard tensor product quadrature to each entry of S_K separately.

A. Standard algorithm

For an arbitrary set of shape functions $\Psi = \{\psi_i \mid i = 1, \ldots, N\}$ on $\mathcal{T}^d$ the standard algorithm for setting up the element stiffness matrix reads:
Algorithm 5.1 (standard algorithm):

1) Choose quadrature rules
$$\mathrm{QR}^{(i)} = S^{(i)} \times W^{(i)} = \{(\eta_0^{(i)}, \omega_0^{(i)}), \ldots, (\eta_{q_i}^{(i)}, \omega_{q_i}^{(i)})\},$$
which incorporate the $\det|D_d'|$ terms of (13) as weight functions (Gauss-Jacobi-Lobatto quadrature) and set
$$\mathrm{QR} = \mathrm{QR}^{(1)} \times \ldots \times \mathrm{QR}^{(d)}.$$
2) Initialize $S_K = 0$
3) For all $(\eta^{(1)}, \ldots, \eta^{(d)}) \in S^{(1)} \times \ldots \times S^{(d)}$ and corresponding weight $(\omega^{(1)}, \ldots, \omega^{(d)})$ do
$$S_K[i][j] += \omega^{(1)} \cdot \ldots \cdot \omega^{(d)} \left(\tilde{\nabla}(\psi_j \circ D_d), \hat{C}\tilde{\nabla}(\psi_i \circ D_d)\right)\Big|_{(\eta^{(1)},\ldots,\eta^{(d)})} \qquad \forall\, 1 \le i, j \le N.$$

B. Sum factorization

Considering Definition 3.4 and Definition 3.5 we observe that after transformation to the reference cube, all of our shape functions $\phi = \psi \circ D_d$ have a common tensor product structure. For $\Psi^{(KS)}$ as well as for $\Psi^{(Lag)}$ we have

$$\Psi \circ D_2 = \Phi = \left\{ \phi_{(B,k_1,k_2)}(\eta_1, \eta_2) = g_{B,k_1}^{(1)}(\eta_1) g_{B,k_1,k_2}^{(2)}(\eta_2) \;\middle|\; \begin{array}{l} 1 \le k_1 \le K_1(B) \\ 1 \le k_2 \le K_2(B, k_1) \end{array} \right\} \tag{13}$$

for $d = 2$ and

$$\Psi \circ D_3 = \Phi = \left\{ \phi_{(B,k_1,k_2,k_3)}(\eta_1, \eta_2, \eta_3) = g_{B,k_1}^{(1)}(\eta_1) g_{B,k_1,k_2}^{(2)}(\eta_2) g_{B,k_1,k_2,k_3}^{(3)}(\eta_3) \;\middle|\; \begin{array}{l} 1 \le k_1 \le K_1(B) \\ 1 \le k_2 \le K_2(B, k_1) \\ 1 \le k_3 \le K_3(B, k_1, k_2) \end{array} \right\} \tag{14}$$

for $d = 3$. Thus, since the components of $\tilde{\nabla}\Phi$ are of the same structure, we obtain by making use of sum factorization ideas the following algorithm of complexity $O(p_K^{2d+1})$ for setting up the element stiffness matrix. For $d = 3$ this algorithm reads:
Algorithm 5.2 (sum factorization 3D):

1) For $i = 1, \ldots, 3$ choose quadrature rules
$$\mathrm{QR}^{(i)} = \{(\eta_{l_i}^{(i)}, \omega_{l_i}^{(i)}) \mid l_i = 0, \ldots, q_i\}$$
which incorporate the $\det|D_3'|$ terms of (13).

2) For all $1 \leq r \leq 3$ and $0 \leq B \leq 13$ let

$$\tilde{\nabla}_r \Phi_B = \Big\{ \tilde{g}^{(1)}_{(B,r,k_1)}(\eta_1)\tilde{g}^{(2)}_{(B,r,k_1,k_2)}(\eta_2)\tilde{g}^{(3)}_{(B,r,k_1,k_2,k_3)}(\eta_3) \ \Big| \ 1 \leq k_1 \leq K_1(B), 1 \leq k_2 \leq K_2(B,k_1), 1 \leq k_3 \leq K_3(B,k_1,k_2) \Big\}$$

3) For $1 \leq r \leq 3$, $0 \leq B \leq 13$, $0 \leq l_i \leq q_i$, $1 \leq k_1 \leq K_1(B)$, $1 \leq k_2 \leq K_2(B,k_1)$, $1 \leq k_3 \leq K_3(B,k_1,k_2)$ compute the auxiliary arrays

$$\begin{aligned} G^{(1)}(B,r,k_1,l_1) &= \tilde{g}^{(1)}_{B,r,k_1}(\eta^{(1)}_{l_1}) \\ G^{(2)}(B,r,k_1,k_2,l_2) &= \tilde{g}^{(2)}_{B,r,k_1,k_2}(\eta^{(2)}_{l_2}) \\ G^{(3)}(B,r,k_1,k_2,k_3,l_3) &= \tilde{g}^{(3)}_{B,r,k_1,k_2,k_3}(\eta^{(3)}_{l_3}) \end{aligned}$$

4) For $1 \leq r, r' \leq 3$ and $0 \leq l_i \leq q_i$ compute the auxiliary array

$$\hat{C}(r',r,l_1,l_2,l_3) = \hat{C}_{(r',r)}(\eta^{(1)}_{l_1}, \eta^{(2)}_{l_2}, \eta^{(3)}_{l_3})$$

5) Initialize $S_K = 0$
6) For $1 \leq r, r' \leq 3$ and $0 \leq B, B' \leq 13$ compute:

$$H^{(1)}[k_1,k_1',l_3,l_2] = \sum_{l_1=0}^{q_1} G^{(1)}(B,r,k_1,l_1)G^{(1)}(B',r',k_1',l_1)\hat{C}(r',r,l_1,l_2,l_3)\omega^{(1)}_{l_1},$$

$$H^{(2)}[k_1,k_1',k_2,k_2',l_3] = \sum_{l_2=0}^{q_2} G^{(2)}(B,r,k_1,k_2,l_2)G^{(2)}(B',r',k_1',k_2',l_2)H^{(1)}[k_1,k_1',l_3,l_2]\omega^{(2)}_{l_2},$$

with

$$\begin{aligned} &1 \leq k_1 \leq K_1(B),\ 1 \leq k_2 \leq K_2(B,k_1),\ 1 \leq k_3 \leq K_3(B,k_1,k_2), \\ &1 \leq k_1' \leq K_1(B'), 1 \leq k_2' \leq K_2(B',k_1'), 1 \leq k_3' \leq K_3(B',k_1',k_2'),\ 0 \leq l_i \leq q_i. \end{aligned}$$

$$S_K[(B,k_1,k_2,k_3)][(B',k_1',k_2',k_3')]+= \sum_{l_3=0}^{q_3} G^{(3)}(B,r,k_1,k_2,k_3,l_3)G^{(3)}(B',r',k_1',k_2',k_3',l_3)H^{(2)}[k_1,k_1',k_2,k_2',l_3]\omega^{(3)}_{l_3},$$

for

$$\begin{aligned} &1 \leq k_1 \leq K_1(B),\ 1 \leq k_2 \leq K_2(B,k_1),\ 1 \leq k_3 \leq K_3(B,k_1,k_2), \\ &1 \leq k_1' \leq K_1(B'), 1 \leq k_2' \leq K_2(B',k_1'), 1 \leq k_3' \leq K_3(B',k_1',k_2'). \end{aligned}$$

Remark 5.3: The 2-dimensional version of Algorithm 5.2 can be obtained by the same ideas as in the 3-dimensional case.
Remark 5.4: For each pair (B, B') a separate quadrature rule could be chosen. In particular for blocks with low polynomial degree this might lead to further savings.
Remark 5.5: The precomputing of the shape functions and coefficient matrix in step 3 and 4 is done due to the fact that evaluations of the shape functions and coefficient matrix can be very expensive, especially for large polynomial degrees or coefficients with a complicated structure. The precomputations of step 3 and 4 lead to a considerable speed-up, since we have to evaluate the shape functions and coefficient matrix just once for all quadrature points.
Remark 5.6: It is not necessary to perform the precomputations of step 3 for each element of a meshing $\mathcal{T}$. Provided the quadrature rules depend only on the internal polynomial degree and due to the fact that $p_e, p_f \leq p_K$ for all edges and faces of the element K, it suffices to compute the arrays of step 3 just once for each $p_K \in \{1, \ldots, p_{max}\}$ and assumed uniform polynomial degree distribution, that is $p_e = p_f = p_k$ for all edges and faces.

C. Spectral Galerkin method

In the last subsection we already exploited the tensor product structure of the shape functions. If, however, the shape functions and the quadrature rules are adapted to each other, then a further reduction of the complexity is possible. To that end we consider in the following quadrature rules of the form

$$\mathrm{QR}^i = S^{(i)} \times W^{(i)} = \{(\eta^{(i)}_0, \omega^{(i)}_0), \ldots, (\eta^{(i)}_{q_i}, \omega^{(i)}_{q_i})\}, \qquad \mathrm{QR} = \mathrm{QR}^1 \times \ldots \times \mathrm{QR}^d$$

which incorporate the $\det |D_d'|$ terms of (13) in conjunction with the modified shape functions $\Phi^{(Lag)}$, where the nodal sets $N^{(i)}$ are subsets of the quadrature points, that is $N^{(i)} \subset S^{(i)}$. Considering the shape functions of $\Phi^{(Lag)}$, we additionally observe a simpler tensor product stucture as in the general cases (13) and (14)

$$\Phi^{(Lag)} = \Big\{ \phi_{(B,k_1,k_2)}(\eta) = g^{(1)}_{B,k_1}(\eta_1) g^{(2)}_{B,k_2}(\eta_2) \ | \ 1 \leq k_i \leq K_i(B) \Big\}$$

for $d = 2$ and

$$\Phi_{13}^{(Lag)} = \left\{ \phi_{(13,k_1,k_2,k_3)}(\eta) = g_{13,k_1}^{(1)}(\eta_1) g_{13,k_2}^{(2)}(\eta_2) g_{13,k_3}^{(3)}(\eta_3) \mid 1 \leq k_i \leq K \right\}$$

for $d = 3$. Evaluating the gradient of the interior bubble shape functions at the quadrature points, we obtain, due to the adaption of shape functions and quadrature rules, a considerable number of zeros. Thus, we can replace the sums in Algorithm 5.2 by the following pattern:

$$\sum_{l_1=0}^{q_1} \tilde{g}_{(B,r,k_1)}^{(1)} \tilde{g}_{(B',r',k_1')}^{(1)} \hat{C}_{r',r} \Big|_{(\eta_{l_1}^{(1)}, \eta_{l_2}^{(2)}, \eta_{l_3}^{(3)})} \to \sum_{l_i \in NZ_{(r,r')}^{(1)}[k_1,k_1']} \tilde{g}_{(B,r,k_1)}^{(1)} \tilde{g}_{(B',r',k_1')}^{(1)} \hat{C}_{r',r} \Big|_{(\eta_{l_1}^{(1)}, \eta_{l_2}^{(2)}, \eta_{l_3}^{(3)})},$$

where the sets of relevant indices

$$NZ_{(r,r')}^{(1)}[k_1, k_1'] := \{ l_1 \in \{0, \ldots, q^{(1)}\} \mid \tilde{g}_{(B,r,k_1)}^{(1)} \tilde{g}_{(B',r',k_1')}^{(1)} \Big|_{\eta_{l_1}^{(1)}} \neq 0 \},$$

can be precomputed easily. (For the summations over l_2 and l_3 we proceed analogously.) Moreover, due to simpler stucture of $\Phi^{(Lag)}$, we are able to permute the summation order of l_i, $i = 1, \ldots, d$ arbitrarily for $B = B' = 13$ in the 3-dimensional case and for all pairings (B, B') in the 2-dimensional case. Since we know the number the none-zero elements $\#NZ_{(r,r')}^{(1)}[k_1, k_1']$, we can estimate the work for setting up the auxiliary arrays H as well as for setting up the stiffness matrix S_K for each permutation of the summation order and choose, of course, the cheapest one. Thus, we can replace step 6 of Algorithm 5.2 for $B = B' = 13$ in 3D and for all pairings B, B' in 2D by the following:[1]
Algorithm 5.7 (spectral Galerkin 3D):

6. For $B = B' = 13$ and all $1 \leq r, r' \leq 3$ do:

1) For $i = 1, ..., 3$, $1 \leq k_i \leq K_i(B)$, $1 \leq k_i' \leq K_i(B')$ and $0 \leq l_i \leq q_i$ compute:
$$F^{(i)}[k_i, k_i', l_i] := G^{(i)}(B, r, k_i, l_i) G^{(i)}(B', r', k_i', l_i),\ NZ^{(i)}[k_i, k_i'] := \{ l_i \mid F^{(i)}[k_i, k_i', l_i] \neq 0 \},\ S_i := \sum_{k_i, k_i'} |NZ^i[k_i, k_i']|$$
2) For all permutations (i_1, i_2, i_3) of $\{1, 2, 3\}$ estimate the work for the summation order $\sum_{l_{i_1}=0}^{q_{i_1}} \sum_{l_{i_2}=0}^{q_{i_2}} \sum_{l_{i_3}=0}^{q_{i_3}}$:
$$\begin{aligned} W_{(i_1,i_2,i_3)} =& (q_{i_1}+1)(q_{i_2}+1) S_{i_3} + && \text{\%setting up } H^{(1)} \\ & K_{i_3}(B) K_{i_3}(B')(q_{i_1}+1) S_{i_2} + && \text{\%setting up } H^{(2)} \\ & K_{i_3}(B) K_{i_3}(B') K_{i_2}(B) K_{i_2}(B') S_{i_1} && \text{\%setting up } S_K \end{aligned}$$
3) Find a permutation (i_1, i_2, i_3) with $W_{(i_1,i_2,i_3)} \leq W_{(i_1',i_2',i_3')}$ for all (i_1', i_2', i_3').
4) For $1 \leq k_i \leq K_i(B)$, $1 \leq k_i' \leq K_i(B')$ and $0 \leq l_i \leq q_i$ compute the auxiliary arrays
$$H^{(1)}[k_{i_3}, k_{i_3}', l_{i_1}, l_{i_2}] = \sum_{l_{i_3} \in NZ^{(i_3)}} F^{(i_3)}[k_{i_3}, k_{i_3}', l_{i_3}] \hat{C}(r', r, l_1, l_2, l_3) \omega_{l_{i_3}}^{(i_3)}$$
$$H^{(2)}[l_{i_1}, k_{i_3}, k_{i_3}', k_{i_2}, k_{i_2}'] = \sum_{l_{i_2} \in NZ^{(i_2)}} F^{(i_2)}[k_{i_2}, k_{i_2}', l_{i_2}] H^{(1)}[l_{i_1}, l_{i_2}, k_{i_3}, k_{i_3}'] \omega_{l_{i_2}}^{(i_2)}$$
5) For all $1 \leq k_i \leq K_i(B)$, $1 \leq k_i' \leq K_i(B')$ add
$$S_K[(B, k_1, k_2, k_3)][(B', k_1', k_2', k_3')] \quad += \sum_{l_{i_1} \in NZ^{(i_1)}} F^{(i_1)}[k_{i_1}, k_{i_1}', l_{i_1}] H^{(2)}[l_{i_1}, k_{i_3}, k_{i_3}', k_{i_2}, k_{i_2}'] \omega_{l_{i_1}}^{(i_1)}$$

Assuming quadradure rules of order

$$q_i = \begin{cases} p_K + q & : \ d = 2,\ i = 1 \\ p_K - 1 + q & : \ d = 2,\ i = 2 \\ p_K + q & : \ d = 3 \end{cases} \tag{15}$$

with $q \geq 0$ and $q = O(1)$, we obtain, completely analogously to [7], a complexity of $O(p_K^{2d})$ for setting up the element stiffness matrix S_K. Thus, asymptotically Algorithm 5.7 is superior to Algorithm 5.2 if we consider the computing time for setting up S_K. However, the critical point is that due to the increased number of internal shape functions for large polynomial degrees p_K the advantage in setting up the stiffness matrix will be offset if we consider an hp-implementation making use of static condensation, since, at least asymptotically, the cost of static condensation dominates the total cost per element. Only numerical tests, which we present below, can tell whether there exists a range $\{p_0, \ldots, p_N\}$ of polynomial degrees where it is preferable to use Algorithm 5.7.

[1]again, we only consider the 3D version

VI. Remarks on static condensation and precomputed arrays

A. Static Condensation

In hp-FEM it is customary to perform static condensation. The partition of the shape functions into external E={vertex, edge, face} and I=internal shape functions implies a corresponding block structure of the local element stiffness matrices S_K as well as of the global stiffness matrix S_{glob}. That is:

$$S_K = \begin{bmatrix} S_K^{EE} & S_K^{EI} \\ S_K^{IE} & S_K^{II} \end{bmatrix} \qquad S_{glob} = \begin{bmatrix} S^{EE} & S^{EI} \\ S^{IE} & S^{II} \end{bmatrix}.$$

Due to the support properties of the internal shape functions, the matrix S^{II} is block diagonal with $S^{II} = diag(S_K^{II})$. Thus, the idea of static condensation is to obtain a drastically reduction of unknowns by elimination of the internal shape functions via the Schur complement $S^c = S^{EE} - S^{EI}(S^{II})^{-1}S^{IE}$, which in turn can be obtained by assembling the condensed element stiffness matrices $S_K^c = S_K^{EE} - S_K^{EI}(S_K^{II})^{-1}S_K^{IE}$. The local static condensation, i.e., computing S_K^c, is an $O(p_K^{3d})$ process, but can be performed by using highly optimized Lapack routines, namely, the routine 'dposv' to solve the linear system of equations $S_K^{II}X = S_K^{IE}$ and the routine 'dgemm' to compute $S_K^c = S_K^{EE} - S_K^{EI}X$. Due to the high degree of optimization of these Lapack routine, we expect this $O(p_K^{3d})$ term to dominate the total cost per element only for large polynomial degrees p_K.

B. Precomputed Arrays

A standard device for computing element stiffness matrices in hp-FEM is to employ *precomputed arrays,* which can lead to considerable savings. To describe this technique, suppose that the matrix A of (1) is constant for each element and that the mesh consists of affine elements only, i.e., all element maps $F_K : \widehat{K} \to K$ of the mesh are affine. Then all element stiffness matrices S_K are given by

$$(S_K)_{ij} = \int_{\widehat{K}} \nabla\psi_j (F_K')^{-1} A (F_K')^{-\top} \nabla\psi_i |\det F_K'| \mathrm{d}\Omega.$$

Since the element map F_K is affine and the matrix A is constant, we note that $(S_K)_{ij}$ can be obtained as a linear combination of the values $\int_{\widehat{K}} \partial_k\psi_j\partial_l\psi_i \mathrm{d}\Omega$. Thus, if the polynomial degree p is the same for all elements of the mesh (and the polynomial basis Ψ is the same for all elements), then an array with values $H_{pre}(k,l,i,j) := \int_{\widehat{K}} \partial_k\psi_j\partial_l\psi_i \mathrm{d}\Omega$ may be precomputed once, and all element stiffness matrices S_K are easily obtained as linear combinations of entries of H_{pre}. If the polynomial degree is not constant on the mesh, then the method of precomputed arrays is still applicable provided that H_{pre} includes all combinations of shape functions needed. In practice, this can be achieved efficiently if the polynomial basis employed is hierarchic, i.e., if the set Ψ_p of shape functions for polynomial degree p is contained in $\Psi_{p'}$ for $p' \geq p$; in that case, the cost of setting up H_{pre} depends only on the maximal polynomial degree of the mesh.

VII. Matrix vector multiplication without setting up the element stiffness matrix

In the last sections we discussed different methods for setting up the element stiffness matrices S_K. However, for solving the final global linear system of equations by an iterative solver such as, for example, the conjugated gradient method, we actually do not need to generate the stiffness matrices explicitly. Instead, it suffices to realize the matrix-vector multiplication $w \mapsto S_{glob}w$. In fact, for the shape functions considered here an "assembly on the fly" is very simple to realize so that we may restrict our attention the elementwise matrix-vector multiplication, i.e., the map $v \mapsto b := S_K v$. In this section we show how to realize such a matrix vector multiplication without setting up the matrix S_K explicitly. For the following we always assume (15) .

A. Sum factorization

We start with a sum factorization idea leading to an algorithm of complexity $O(p_K^{d+1})$ for performing one multiplication. Since the cases $d = 2$ and $d = 3$ are very similar, we will describe this idea only for the case $d = 3$. On the reference tetrahedron $\mathcal{T}^3$ let the shape functions Ψ be given by Definition 3.5. That is

$$\psi_{(B,k_1,k_2,k_3)} = \phi_{(B,k_1,k_2,k_3)} \circ D_3^{-1} \quad \text{with} \quad \phi_{(B,k_1,k_2,k_3)}(\eta_1,\eta_2,\eta_3) = g^{(1)}_{(B,k_1)}(\eta_1) g^{(2)}_{(B,k_1,k_2)}(\eta_2) g^{(3)}_{(B,k_1,k_2,k_3)}(\eta_3)$$

and

$$0 \leq B \leq 13, \quad 1 \leq k_1 \leq K_1(B), \quad 1 \leq k_2 \leq K_2(B,k_1), \quad 1 \leq k_3 \leq K_3(B,k_1,k_2).$$

Applying the quadrature rule

$$\mathrm{QR} = \mathrm{QR}^1 \times \mathrm{QR}^2 \times \mathrm{QR}^3 \ \text{ with } \qquad \mathrm{QR}^i = S^{(i)} \times W^{(i)} = \{(\eta_0^{(i)}, \omega_0^{(i)}), \ldots, (\eta_{q_i}^{(i)}, \omega_{q_i}^{(i)})\},$$

on $\mathcal{Q}^3$, with the abbreviations $I = (B, k_1, k_2, k_3)$ and $I' = (B', k_1', k_2', k_3')$, the entries of the element stiffness matrix S_K are given by (see Algorithm 5.1)

$$S_K = \left[\sum_{l_1,l_2,l_3} \sum_{r,r'=1}^{3} \omega_{l_1}^{(1)} \omega_{l_2}^{(2)} \omega_{l_3}^{(3)} \left(\tilde{\nabla}_{r'} \phi_{I'} \hat{C}_{r'r} \tilde{\nabla}_r \phi_I \right) \Big|_{(\eta_{l_1}^{(1)}, \eta_{l_2}^{(2)}, \eta_{l_3}^{(3)})} \right]_{I,I'}.$$

Thus, the vector $b := S_K v$ can be evaluated as

$$b_I = \sum_{(r,r',B',k_1',k_2',k_3')} \sum_{(l_1,l_2,l_3)} \omega_{l_1}^{(1)} \omega_{l_2}^{(2)} \omega_{l_3}^{(3)} \left(\tilde{\nabla}_{r'} \phi_{I'} \hat{C}_{r',r} \tilde{\nabla}_r \phi_I \right) \Big|_{(\eta_{l_1}^{(1)}, \eta_{l_2}^{(2)}, \eta_{l_3}^{(3)})} v_{I'} \tag{16}$$

and following algorithm realizes a sum factorization idea leading to an efficient matrix vector multiplication without setting up the element stiffness matrix S_K and complexity $O(p_K^{d+1})$.

Algorithm 7.1 (Matrix vector multiplication on the fly - 3D):

1) For $i = 1, \ldots, 3$ choose quadrature rules

$$\mathrm{QR}^{(i)} = \{(\eta_{l_i}^{(i)}, \omega_{l_i}^{(i)}) \mid l_i = 0, \ldots, q_i)\},$$

which incorporate the $\det|D_3'|$ terms of (13).

2) For all $1 \le r \le 3$ and $0 \le B \le 13$ let

$$\tilde{\nabla}_r \Phi_B = \Big\{ \tilde{g}_{(B,r,k_1)}^{(1)}(\eta_1) \tilde{g}_{(B,r,k_1,k_2)}^{(2)}(\eta_2) \tilde{g}_{(B,r,k_1,k_2,k_3)}^{(3)}(\eta_3) \ \Big| \ 1 \le k_1 \le K_1(B), 1 \le k_2 \le K_2(B,k_1), 1 \le k_3 \le K_3(B,k_1,k_2) \Big\}.$$

3) For $1 \le r \le 3$, $0 \le B \le 13$, $0 \le l_i \le q_i$, $1 \le k_1 \le K_1(B)$, $1 \le k_2 \le K_2(B,k_1)$, $1 \le k_3 \le K_3(B,k_1,k_2)$ compute the auxiliary arrays

$$\begin{aligned} G^{(1)}(B,r,k_1,l_1) &= \tilde{g}_{B,r,k_1}^{(1)}(\eta_{l_1}^{(1)}) \\ G^{(2)}(B,r,k_1,k_2,l_2) &= \tilde{g}_{B,r,k_1,k_2}^{(2)}(\eta_{l_2}^{(2)}) \\ G^{(3)}(B,r,k_1,k_2,k_3,l_3) &= \tilde{g}_{B,r,k_1,k_2,k_3}^{(3)}(\eta_{l_3}^{(3)}). \end{aligned}$$

4) For $1 \le r, r' \le 3$, and $0 \le l_i \le q_i$ compute the auxiliary array

$$\hat{C}(r', r, l_1, l_2, l_3) = \hat{C}_{(r',r)}(\eta_{l_1}^{(1)}, \eta_{l_2}^{(2)}, \eta_{l_3}^{(3)}).$$

5) Initialize $b = 0$

6) Compute the auxiliary arrays

$$\begin{aligned} H^{(1)}[r', B', k_1', k_2', l_3] &= \sum_{k_3'} v_{(B',k_1',k_2',k_3')} G^{(3)}(B', r', k_1', k_2', k_3', l_3) \\ H^{(2)}[r', B', k_1', l_2, l_3] &= \sum_{k_2'} H^{(1)}[r', B', k_1', k_2', l_3] G^{(2)}(B', r', k_1', k_2', l_2) \\ H^{(3)}[r', l_1, l_2, l_3] &= \sum_{B',k_1'} H^{(2)}[r', B', k_1', l_2, l_3] G^{(1)}(B', r', k_1', l_1) \\ H^{(4)}[r, B, k_1, l_2, l_3] &= \sum_{r',l_1} \omega_{l_1}^{(1)} H^{(3)}[r', l_1, l_2, l_3] \hat{C}(r', r, l_1, l_2, l_3) G^{(1)}(B, r, k_1, l_1) \\ H^{(5)}[r, B, k_1, k_2, l_3] &= \sum_{l_2} \omega_{l_2}^{(2)} H^{(4)}[r, B, k_1, l_2, l_3] G^{(2)}(B, r, k_1, k_2, l_2) \end{aligned}$$

for $1 \le r, r' \le 3, 0 \le B, B' \le 13, 0 \le l_i \le q_i$ and

$$\begin{aligned} &1 \le k_1 \le K_1(B),\ 1 \le k_2 \le K_2(B,k_1),\ 1 \le k_3 \le K_3(B,k_1,k_2), \\ &1 \le k_1' \le K_1(B'),\ 1 \le k_2' \le K_2(B',k_1'),\ 1 \le k_3' \le K_3(B',k_1',k_2'). \end{aligned}$$

For $0 \le B \le 13,\ 1 \le k_1 \le K_1(B),\ 1 \le k_2 \le K_2(B,k_1),\ 1 \le k_3 \le K_3(B,k_1,k_2)$ compute

$$b_{(B,k_1,k_2,k_3)} = \sum_{l_3} \omega_{l_3}^{(3)} H^{(5)}[r, B, k_1, k_2, l_3] G^{(3)}(B, r, k_1, k_2, k_3, l_3).$$

The algorithm can be divided into two parts, namely, steps 1-4 which are the initialization and steps 5,6, which realize the actual matrix vector multiplication. Irrespective of the number of multiplications, the initialization has to be done only once. More precisely, assuming that a finite element mesh $\mathcal{T}$ is given, the computation of the arrays of step 2 has to be done only once for each internal polynomial degree p_K occurring in $\mathcal{T}$ (see Remark 5.6) and the computation of the symmetric auxiliary array $\hat{C}(r', r, l_1, l_2, l_3)$ with its $(1/2)3^2(q_1+1)(q_2+1)(q_3+1)$ entries has to be done once for each $K \in \mathcal{T}$. Even for elements with large polynomial degrees and resulting quadrature rules of high order this additional storage is of an acceptable size. For example, for an element with $p_K = q_i = 10$ the storage of $\hat{C}(r', r, l_1, l_2, l_3)$ requires, assuming 8 Byte per entry, about 47 KByte. Consequently, considering the computing time for one matrix vector multiplication, we will omit the time necessary for steps 1-4. For the modified shape functions Φ^{Lag} we obtain further savings by adding up only the non-zero elements in the sums of step 6.

Remark 7.2: The disadvantage of adding up only the non-zero elements is a frequent accessing of non-contiguous pieces of computer memory, which may offset some of these gains on some memory architectures.

Figures 8, 9 and Tables I-IV show the computing time for the matrix vector multiplication for different sets of shape functions and different methods. Note that for a matrix vector multiplications performed with the blas-routine we additionally have to set up the stiffness matrix S_K.

Algorithm 7.1 realizes one special summation order and the question arises if there are other good summation orders. For the case of ϕ^{KS} the following lemma answers this question.

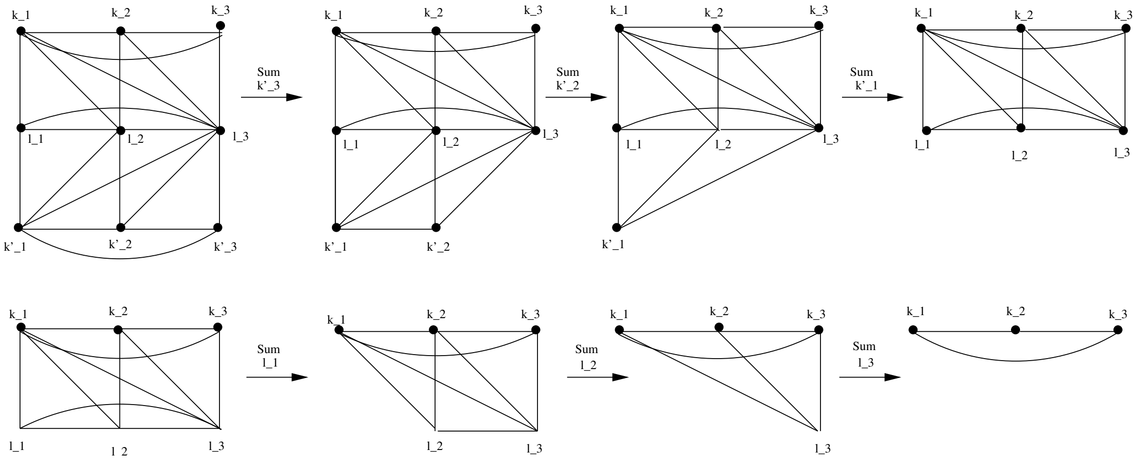
Lemma 7.3: Let the shape functions ψ^{KS} be given by Definition 3.4 or Definition 3.5 respectively. Then the only summation order for (16) yielding a complexity $O(p^{d+1})$ for one matrix vector multiplication is given by $(l_3, l_2, l_1, k_1', k_2', k_3')$ for $d = 3$ or (l_2, l_1, k_1', k_2') for $d = 2$ respectively. All other summation orders lead to a complexity worse than $O(p^{d+1})$.

Proof: Since the proofs for $d = 2$ and $d = 3$ are completely analogous, we will only prove the 3-dimensional case. We consider only the internal shape functions (B=B'=13). The upper left graph in Figure 2 shows the dependences between the entities $k_1, k_2, k_3, l_1, l_2, l_3, k_1', k_2', k_3'$. Each vertex represents one of those entities and an edge between two vertices V_1 and V_2 exists if and only if there is a factor in

$$b_{(B,k_1,k_2,k_3)} = \sum_{\{r,r',B'\}} \sum_{\{k_1',k_2',k_3',l_1,l_2,l_3\}} \omega_{l_1}^{(1)} \omega_{l_2}^{(2)} \omega_{l_3}^{(3)} G_{B',r'}^{(3)}(k_1', k_2', k_3', l_3) G_{B',r'}^{(2)}(k_1', k_2', l_2) G_{B',r'}^{(1)}(k_1', l_1) \hat{C}_{r',r}(l_1, l_2, l_3)$$
$$G_{B,r}^{(1)}(k_1, l_1) G_{B,r}^{(2)}(k_1, k_2, l_2) G_{B,r}^{(3)}(k_1, k_2, k_3, l_3) v_{(B',k_1',k_2',k_3')}.$$

which depends on both entities represented by V_1 and V_2. For example, $G_{B',r'}^{(3)}(k_1', k_2', k_3', l_3)$ implies the edges $\{k_1', k_2'\}$, $\{k_1', k_3'\}$, $\{k_1', l_3\}$, $\{k_2', k_3'\}$, $\{k_2', l_3\}$ and $\{k_3', l_3\}$. Now, summation over one entity, characterized by V, means creating an auxiliary array depending on all entities adjacent to V and leads to a new dependence graph. Since for the internal shape functions each k_i, k_i' and l_i covers a range of $O(p_K)$, the total amount of work for such a summation is given by $O(p_K^{1+A(V)})$, where $A(V)$ is the number of vertices adjacent to V. Thus, in order to obtain a complexity of $O(p_K^4)$ or better, we are not allowed to sum over entities with more than 3 adjacent vertices. Figure 2 shows the unique summation order leading to complexity $O(p^4)$. ■

Fig. 2. Proof of Lemma 7.3



Remark 7.4: There exists one exception to Lemma 7.3, namely,

$$b_{(B,k_1,k_2,k_3)} = \sum_{(r,r',B',k_1',k_2',k_3',l_*,l_*)} \sum_{(l_i)} \dots,$$

where we start with summation over l_i. In this case the inner sum is independent of the vector $v_{(B',k_1',k_2',k_3')}$ and can be precomputed as an auxiliary array H. However, since this inner sum contains the factor $\hat{C}(r', r, l_1, l_2, l_3)$ which depends on the element K, we have to compute an auxiliary array $H^{(K)}$ for each element K of a finite element meshing separately. Starting the summation with

- l_1, this leads to
$$H^{(K)}_{r,r',B,B'}[k_1,k'_1,l_2,l_3] = \sum_{l_1} \omega^{(1)}_{l_1} \hat{C}_{r',r}(l_1,l_2,l_3) G^{(1)}_{B,r}(k_1,l_1) G^{(1)}_{B',r'}(k'_1,l_1),$$
- l_2, this leads to
$$H^{(K)}_{r,r',B,B'}[k_1,k'_1,k_2,k'_2,l_1,l_3] = \sum_{l_2} \omega^{(2)}_{l_2} \hat{C}_{r',r}(l_1,l_2,l_3) G^{(2)}_{B,r}(k_1,k_2,l_2) G^{(2)}_{B',r'}(k'_1,k'_2,l_2),$$
- l_3, this leads to
$$H^{(K)}_{r,r',B,B'}[k_1,k'_1,k_2,k'_2,k_3,k'_3,l_1,l_2] = \sum_{l_3} \omega^{(3)}_{l_3} \hat{C}_{r',r}(l_1,l_2,l_3) G^{(3)}_{B,r}(k_1,k_2,k_3,l_3) G^{(3)}_{B',r'}(k'_1,k'_2,k'_3,l_3).$$

As we can see, setting up these auxiliary arrays is of complexity worse than $O(p^{d+1})$ and has to be done for each element of a mesh separately. Thus, for large polynomial degrees and just a few matrix vector multiplications (i.e., good preconditioners) setting up the arrays $H^{(K)}$ can become more time consuming than performing the matrix vector multiplications with use of Algorithm 7.1. Another point is that storing $H^{(K)}_{r,r',B,B'}[k_1,k'_1,l_2,l_3]$ for $p_K = q_i = 10$ requires about $(1/2) * 3^2 * 6^2 * 10^4 * 8$ Byte $\approx$ 17MByte of memory. Consequently, we exclude a summation order starting with l_i.

Remark 7.5: Lemma 7.3 says that $(l_3,l_2,l_1,k'_1,k'_2,k'_3)$ or (l_2,l_1,k'_1,k'_2) respectively are the uniquely determined best choices for a global and constant summation order and one may think of a summation order depending on B and B'. However, for $\Psi^{(KS)}$, due to factors $g^{(j)}_{(B,\dots)}$ which depend several k_i or k'_i, this approach will not be practicable and leads only to an extensive case differentiation. For the modified shape functions $\Psi^{(Lag)}$ we investigate this possibility in the next subsection.

B. *Speeding up the matrix vector multiplication by spectral Galerkin ideas*

In this subsection we want to exploit the special structure of Φ^{Lag} to obtain a speed up for the on the fly matrix vector multiplication. However, since in the 3-dimensional case we only modified the internal shape functions and the internal shape functions take only a fraction of the total computing time (Fig. 8) will restrict our attention to the 2-dimensional case. Thus we have

$$\psi_{(B,k_1,k_2)} = \phi_{(B,k_1,k_2)} \circ D_2^{-1} \text{ with } \phi_{(B,k_1,k_2)}(\eta_1,\eta_2) = g^{(1)}_{(B,k_1)}(\eta_1) g^{(2)}_{(B,k_2)}(\eta_2) \text{ and } 0 \le B \le 5, \quad 1 \le k_i \le K_i(B).$$

The main idea to achieve a speed-up compared to Algorithm 7.1 is to exploit that the shape functions and the quadrature rules are adapted to each together with the simpler structure of the shape functions. Considering

$$b_{(B,k_1,k_2)} = \sum_{\{r,r',B'\}} \sum_{\{k'_1,k'_2,l_1,l_2\}} \omega^{(1)}_{l_1}\omega^{(2)}_{l_2} G^{(2)}_{B',r'}(k'_2,l_2) G^{(1)}_{B',r'}(k'_1,l_1) \hat{C}_{r',r}(l_1,l_2) G^{(1)}_{B,r}(k_1,l_1) G^{(2)}_{B,r}(k_2,l_2) v_{(B',k'_1,k'_2)},$$

with $G^{(i)}_{B,r}(k_i,l_i)$, $\hat{C}_{r',r}(l_1,l_2)$ and $v_{(B',k'_1,k'_2)}$ as in the previous section, we have 24 possible summation orders for $\{k'_1,k'_2,l_1,l_2\}$. Thus, we want to find and apply the cheapest, or at least a good summation order depending on B,B',r,r'. However, some of these permutations can be excluded a priori:

- $(k'_i,l_j,*,*)$ since a calculation reveals that for all $i,j \in \{1,2\}$ $(l_j,k'_i,*,*)$ is equal to or more efficient than $(k'_i,l_j,*,*)$.
- $(k'_i,k'_{\bar{i}},l_j,l_{\bar{j}})$ since this is equivalent to setting up a block of the stiffness matrix.
- $(*,*,k'_i,l_j)$ du to an argumentation as in Remark 7.4.
- $(l_i,k'_{\bar{i}},l_{\bar{i}},k'_i)$ since an efficient implementation becomes equivalent to $(*,*,k'_i,l_j)$.

Thus, it remain four permutations of type $(l_j,l_{\bar{j}},k'_i,k'_{\bar{i}})$ and two permutations of type $(l_j,k'_j,l_{\bar{j}},k'_{\bar{j}})$. Considering (l_1,k'_1,l_2,k'_2), we have

$$b_{(B,k_1,k_2)} = \sum_{l_1} \omega^{(1)}_{l_1} G^{(1)}_{B,r}(k_1,l_1) \sum_{r',B',k'_1} G^{(1)}_{B',r'}(k'_1,l_1) H^{(2)}_{r,r',B,B'}(k_2,k'_1,l_1),$$

where

$$H^{(2)}_{r,r',B,B'}(k_2,k'_1,l_1) = \sum_{l_2} H_{r',B'}(k'_1,l_2) \hat{C}_{r',r}(l_1,l_2) G^{(2)}_{B,r}(k_2,l_2) \tag{17}$$

$$H_{r',B'}(k'_1,l_2) = \sum_{k'_2} \omega^{(2)}_{l_2} G^{(2)}_{B',r'}(k'_2,l_2) v_{(B',k'_1,k'_2)}. \tag{18}$$

Since $G^{(1)}_{B,r}(k_1,l_1)$ depends on the same entities as the auxiliary array $H^{(2)}$, we can avoid setting up the array $H^{(2)}_{r,r',B,B'}(k_2,k'_1,l_1)$ and evaluate the sum with almost the same or even less work as:

$$b_{(B,k_1,k_2)} = \sum_{l_1} \omega^{(1)}_{l_1} G^{(1)}_{B,r}(k_1,l_1) \sum_{r',B',k'_1} \sum_{l_2} G^{(1)}_{B',r'}(k'_1,l_1) H_{r',B'}(k'_1,l_2) \hat{C}_{r',r}(l_1,l_2) G^{(2)}_{B,r}(k_2,l_2),$$

which in turn is equivalent to

$$b_{(B,k_1,k_2)} = \sum_{l_1} \omega^{(1)}_{l_1} G^{(1)}_{B,r}(k_1,l_1) \sum_{l_2} \sum_{r',B',k_1'} G^{(1)}_{B',r'}(k_1',l_1) H_{r',B'}(k_1',l_2) \hat{C}_{r',r}(l_1,l_2) G^{(2)}_{B,r}(k_2,l_2).$$

Thus, since we can argue in the same way for (l_2, k_2', l_1, k_1'), we can restrict our algorithm to consider the four permutations of type $(l_j, l_{\bar{j}}, k_i', k_{\bar{i}}')$.

Algorithm 7.6 (spectral matrix vector multiplication on the fly):

1) For $i = 1, 2$ choose quadrature rules

$$\mathrm{QR}^{(i)} = \{(\eta^{(i)}_{l_i}, \omega^{(i)}_{l_i}) \mid l_i = 0, \ldots, q_i)\}$$

which incorporate the $\det|D_2'|$ term of (13).

2) For all $1 \le r \le 2$ and $0 \le B \le 5$ let

$$\tilde{\nabla}_r \Phi_B = \left\{ \tilde{g}^{(1)}_{(B,r,k_1)}(\eta_1) \tilde{g}^{(2)}_{(B,r,k_2)}(\eta_2) \;\middle|\; 1 \le k_i \le K_i(B) \right\}$$

3) For $1 \le r \le 2$, $0 \le B \le 5$, $0 \le l_i \le q_i$, $1 \le k_i \le K_i(B)$ compute

$$G^{(1)}(B,r,k_1,l_1) = \tilde{g}^{(1)}_{B,r,k_1}(\eta^{(1)}_{l_1}) \qquad NZ^{(1)}(B,r,k_1) = \{l_1 \mid G^{(1)}(B,r,k_1,l_1) \neq 0\}$$
$$G^{(2)}(B,r,k_2,l_2) = \tilde{g}^{(2)}_{B,r,k_2}(\eta^{(2)}_{l_2}) \qquad NZ^{(2)}(B,r,k_2) = \{l_2 \mid G^{(2)}(B,r,k_2,l_2) \neq 0\}$$

4) For $1 \le r, r' \le 3$ and $0 \le l_i \le q_i$ compute the auxiliary array

$$\hat{C}(r',r,l_1,l_2) = \hat{C}_{(r',r)}(\eta^{(1)}_{l_1}, \eta^{(2)}_{l_2})$$

5) Initialize $b = 0$, $H^{(2)}[r',l_2,l_1] = 0$

6) For $1 \le r' \le 2$, $0 \le l_i \le q_i$ compute

$$H^{(2)}[r',l_2,l_1] = \sum_{B',k_1',k_2'} v_{(B',k_1',k_2')} \omega^{(1)}_{l_1} \omega^{(2)}_{l_2} G^{(1)}(B',r',k_1',l_1) G^{(2)}(B',r',k_2',l_2)$$

as follows: For all $1 \le r' \le 2$, $0 \le B' \le 5$ compute

a)

$$S_1 := \sum_{k_1'} \# NZ^{(1)}(B',r',k_1') \qquad S_2 := \sum_{k_2'} \# NZ^{(2)}(B',r',k_2')$$

b) If $[(q_2+1)S_1 + K_1(B')S_2] \le [(q_1+1)S_2 + K_2(B')S_1]$

Initialize $H^{(1)}[k_1',l_2] = 0$

Add $H^{(1)}[k_1',l_2] += v_{(B',k_1',k_2')} \omega^{(2)}_{l_2} G^{(2)}(B',r',k_2',l_2)$

for all $1 \le k_1' \le K_1(B')$, $1 \le k_2' \le K_2(B')$, $l_2 \in NZ^{(2)}(B',r',k_2)$.

Add $H^{(2)}[r',l_2,l_1] += \omega^{(1)}_{l_1} H^{(1)}[k_1',l_2] G^{(1)}(B',r',k_1',l_1)$

for all $1 \le k_1' \le K_1(B')$, $l_1 \in NZ^{(1)}(B',r',k_1)$, $0 \le l_2 \le q_2$.

c) If $[(q_2+1)S_1 + K_1(B')S_2] > [(q_1+1)S_2 + K_2(B')S_1]$

Initialize$H^{(1)}[k_2',l_1] = 0$

Add $H^{(1)}[k_2',l_1] += v_{(B',k_1',k_2')} \omega^{(1)}_{l_1} G^{(1)}(B',r',k_1',l_1)$

for all $1 \le k_1' \le K_1(B')$, $1 \le k_2' \le K_2(B')$, $l_1 \in NZ^{(1)}(B',r',k_1)$.

Add $H^{(2)}[r',l_2,l_1] += \omega^{(2)}_{l_2} H^{(1)}[k_2',l_1] G^{(2)}(B',r',k_2',l_2)$

for all $1 \le k_1' \le K_1(B')$, $l_1 \in NZ^{(1)}(B',r',k_1)$, $0 \le l_2 \le q_2$.

7) Compute

$$b_{(B,k_1,k_2)} = \sum_{(r,r',l_1,l_2)} H^{(2)}[r',l_2,l_1] \hat{C}(r',r,l_1,l_2) G^{(1)}(I_1) G^{(2)}(I_2)$$

as follows: For all $1 \le r \le 2$, $0 \le B \le 5$ compute

a)

$$S_1 := \sum_{k_1} \# NZ^{(1)}(B,r,k_1) \qquad S_2 := \sum_{k_2} \# NZ^{(2)}(B,r,k_2)$$

b) If $[2(q_2+1)S_1+K_1(B)S_2] \leq [2(q_1+1)S_2+K_2(B)S_1]$

Initialize$H^{(3)}[k_1,l_2]=0$

Add $\quad H^{(3)}[k_1,l_2]+=\hat{C}(r',r,l_1,l_2)H^{(2)}[r',l_2,l_1]G^{(1)}(B,r,k_1,l_1)$

for all $1 \leq k_1 \leq K_1(B)$, $l_1 \in NZ^{(1)}(B,r,k_1)$, $0 \leq r' \leq 2$, $0 \leq l_2 \leq q_2$.

Add $\quad b_{(B,k_1,k_2)}+=H^{(3)}[k_1,l_2]G^{(2)}(B,r,k_2,l_2)$

for all $1 \leq k_2 \leq K_2(B)$, $l_2 \in NZ^{(2)}(B,r,k_2)$, $1 \leq k_1 \leq K_1(B)$.

c) If $[2(q_2+1)S_1+K_1(B)S_2] > [2(q_1+1)S_2+K_2(B)S_1]$

Initialize$H^{(3)}[k_2,l_1]=0$

Add $\quad H^{(3)}[k_2,l_1]+=\hat{C}(r',r,l_1,l_2)H^{(2)}[r',l_2,l_1]G^{(2)}(B,r,k_2,l_2)$

for all $1 \leq k_2 \leq K_2(B)$, $l_2 \in NZ^{(2)}(B,r,k_2)$, $0 \leq r' \leq 2$, $0 \leq l_1 \leq q_1$.

Add $\quad b_{(B,k_1,k_2)}+=H^{(3)}[k_2,l_1]G^{(1)}(B,r,k_1,l_1)$

for all $1 \leq k_1 \leq K_1(B)$, $l_1 \in NZ^{(1)}(B,r,k_1)$, $1 \leq k_2 \leq K_2(B)$.

The basic idea of this algorithm is to set up the auxiliary array

$$H^{(2)}[r',l_2,l_1]=\sum_{B'}\sum_{\{k_1',k_2'\}} v_{(B',k_1',k_2')}\omega_{l_1}^{(1)}\omega_{l_2}^{(2)}G^{(1)}(B',r',k_1',l_1)G^{(2)}(B',r',k_2',l_2)$$

and the vector

$$b_{(B,k_1,k_2)}=\sum_{\{r,r'\}}\sum_{\{l_1,l_2\}} H^{(2)}[r',l_2,l_1]\hat{C}(r',r,l_1,l_2)G^{(1)}(B,r,k_1,l_1)G^{(2)}(B,r,k_2,l_2)$$

by applying the more efficient summation order of $\{k_1',k_2'\}$ or $\{l_1,l_2\}$ respectively depending on (B',r') and (B,r). Since we always have to add up only the non-zero terms, one matrix vector multiplication has a total amount of work given by:

$$W_{Mv}=\sum_{B,r}W_b(B,r)+\sum_{B',r'}W_H(B',r'),$$

with

$$\begin{aligned} W_H(B',r') &= \min\{\ (q_2+1)S_1(B',r')+K_1(B')S_2(B',r'),(q_1+1)S_2(B',r')+K_2(B')S_1(B',r')\ \}, \\ W_b(B,r) &= \min\{2(q_2+1)S_1(B,r)+K_1(B)S_2(B,r),2(q_1+1)S_2(B,r)+K_2(B)S_1(B,r)\ \} \end{aligned}$$

and $S_i(B,r)=\sum_{k_i} NZ^{(i)}(B,r,k_i)$, $S_i(B',r')=\sum_{k_i'} NZ^{(i)}(B',r',k_i')$. The complexity of Algorithm 7.6 is still $O(p^3)$ but it reduces the computing time significantly. (see Figure 5 and Tables I, II)

VIII. Collection of numerical results

This section collects all numerical results for the 2- and 3-dimensional case. For our computations we chose $A(x)=I$ but proceed as in the case of non-constant coefficients. The quadrature rules which we use are Gauss-Lobatto-Jacobi rules

$$\mathrm{QR}=\mathrm{QR}^1\times\ldots\times\mathrm{QR}^d \qquad \text{with} \qquad \mathrm{QR}^i=S^{(i)}\times W^{(i)}=\{(\eta_0^{(i)},\omega_0^{(i)}),\ldots,(\eta_{q_i}^{(i)},\omega_{q_i}^{(i)})\},$$

which incorporate the $|\det D_d'|$ terms. That is, we have the following weight functions:

$$\omega=1 \text{ for } \mathrm{QR}^1, \quad \omega=(1-\eta_2) \text{ for } \mathrm{QR}^2, \quad \omega=(1-\eta_3)^2 \text{ for } \mathrm{QR}^3\,.$$

For the 2-dimensional case we consider a quadrature order of $q_1=q_2=p_K$ and define the following 3 types of quadrature for the 3-dimensional case:

Typ-1: $q_1=q_2=q_3=p_K$, Typ-2: $q_1=q_2=q_3=p_K+1$, Typ-3: $q_1=q_2=q_3=p_K+2$.

Remark 8.1: The quadrature of Typ-1 and the quadrature we use in the 2-dimensional case are not of the minimal possible order. Indeed, setting up the stiffness matrix in 2D could also be done with $q_1=p_K$, $q_2=p_K-1$.

The lagrange shape functions $\Phi^{(Lag)}$ are always adapted to the quadrature and we assume a uniform polynomial degree distribution on K. That is, all edges and faces of K have the same polynomial degree p_K as K has. All computations are executed on the same computer, a Pentium IV, 2400 MHz with 1GB main memory. Tables I- IV show the computing time for the generation of the stiffness matrix (gen), for performing the static condensation (sc) and for one matrix vector multiplication (Mv). Furthermore, labeling the tables and figures we use the following abbreviations:

KS	-	our computation is performed with the shape functions $\Phi^{(KS)}$
Lag	-	our computation is performed with the shape functions $\Phi^{(Lag)}$
blas	-	our computation is performed with BLAS or LAPACK routines
sum fact.	-	our computation is performed with the sum factorization algorithm
spect Gal.	-	our computation is performed with the spectral Galerkin algorithm

We denote the total number of shape functions by DOF and the number of internal shape functions by INT.

TABLE I

KS - COMPUTING TIME - QUADRATURE: $q_i = p_K$ - 2D

p_K	DOF	INT	gen	sc	S*v (blas)	S*v (sum fact)
4	15	3	1.98e-04	3.70e-05	3.00e-06	3.50e-05
5	21	6	2.97e-04	3.10e-05	4.00e-06	4.00e-05
6	28	10	4.56e-04	5.00e-05	4.00e-06	5.60e-05
7	36	15	6.95e-04	6.10e-05	8.00e-06	7.00e-05
8	45	21	1.10e-03	9.30e-05	9.00e-06	9.40e-05
9	55	28	1.58e-03	1.53e-04	1.20e-05	1.23e-04
10	66	36	2.29e-03	2.35e-04	1.40e-05	1.57e-04
11	78	45	3.29e-03	3.60e-04	1.90e-05	1.97e-04
12	91	55	4.62e-03	5.48e-04	2.50e-05	2.36e-04
13	105	66	6.40e-03	8.55e-04	3.70e-05	2.88e-04
14	120	78	8.64e-03	1.22e-03	4.20e-05	3.44e-04
15	136	91	1.13e-02	1.75e-03	5.90e-05	4.20e-04
20	231	171	4.02e-02	7.79e-03	2.10e-04	9.10e-04
25	351	276	1.00e-01	2.54e-02	4.04e-04	1.69e-03
30	496	406	2.27e-01	7.57e-02	6.89e-04	2.75e-03
50	1326	1176	2.27e+00	1.23e+00	4.49e-03	1.20e-02

TABLE II

LAG - COMPUTING TIME - QUADRATURE: $q_i = p_K$ - 2D

p_K	DOF	INT	gen	sc	S*v (blas)	S*v (sum fact)	S*v (spect Gal)
4	18	6	2.18e-04	9.60e-05	2.00e-06	3.40e-05	2.30e-05
5	27	12	2.96e-04	1.16e-04	4.00e-06	4.10e-05	2.80e-05
6	38	20	4.30e-04	1.72e-04	4.00e-06	5.30e-05	4.90e-05
7	51	30	6.09e-04	2.55e-04	8.00e-06	6.70e-05	4.40e-05
8	66	42	8.62e-04	3.35e-04	1.80e-05	8.80e-05	5.30e-05
9	83	56	1.14e-03	5.58e-04	2.70e-05	1.14e-04	6.30e-05
10	102	72	1.62e-03	9.30e-04	3.60e-05	1.54e-04	7.70e-05
11	123	90	2.09e-03	1.51e-03	4.60e-05	2.03e-04	9.40e-05
12	146	110	2.81e-03	2.36e-03	6.40e-05	2.42e-04	1.11e-04
13	171	132	3.54e-03	3.54e-03	9.10e-05	3.01e-04	1.31e-04
14	198	156	4.63e-03	5.26e-03	1.25e-04	3.58e-04	1.53e-04
15	227	182	5.70e-03	7.51e-03	1.75e-04	4.27e-04	1.81e-04
20	402	342	1.66e-02	3.67e-02	5.32e-04	9.00e-04	3.61e-04
30	902	812	7.74e-02	4.03e-01	2.08e-03	2.69e-03	9.70e-04
50	2502	2352	8.43e-01	7.31e+00	1.51e-02	1.15e-02	3.98e-03

TABLE III

KS - COMPUTING TIME - QUADRATURE: $q_i = p_K$ - 3D

p_K	DOF	INT	gen	sc	S*v (blas)	S*v (sum fact)
4	35	1	6.00e-03	1.43e-04	5.00e-06	7.65e-04
5	56	4	1.20e-02	2.76e-04	9.00e-06	1.20e-03
6	84	10	2.25e-02	6.24e-04	1.90e-05	1.81e-03
7	120	20	3.87e-02	9.92e-04	3.60e-05	3.23e-03
8	165	35	7.25e-02	2.45e-03	6.90e-05	4.35e-03
9	220	56	1.29e-01	5.57e-03	1.25e-04	6.92e-03
10	286	84	2.14e-01	1.16e-02	2.34e-04	9.47e-03
11	364	120	4.02e-01	2.48e-02	4.11e-04	1.28e-02
12	455	165	5.22e-01	5.00e-02	6.36e-04	1.63e-02
13	560	220	7.90e-01	9.71e-02	9.23e-04	2.22e-02
14	680	286	1.12e+00	1.83e-01	1.34e-03	2.78e-02
15	816	364	1.60e+00	3.37e-01	1.88e-03	3.61e-02
20	1771	969	8.21e+00	3.98e+00	9.07e-03	9.78e-02
25	3276	2024	2.80e+01	2.47e+01	3.08e-02	2.17e-01
30	5456	3654	7.59e+01	1.10e+02	8.52e-02	4.19e-01

TABLE IV
LAG - COMPUTING TIME - QUADRATURE: $q_i = p_K$ - 3D

p_K	DOF	INT	gen	sc	S*v (blas)	S*v (sum)
4	35	1	5.99e-03	1.35e-04	4.00e-06	7.69e-04
5	60	8	1.18e-02	3.30e-04	1.00e-05	1.21e-03
6	101	27	2.27e-02	1.03e-03	2.70e-05	1.97e-03
7	164	64	4.13e-02	3.29e-03	7.10e-05	3.03e-03
8	255	125	8.02e-02	1.12e-02	1.81e-04	4.37e-03
9	380	216	1.45e-01	3.44e-02	4.42e-04	6.75e-03
10	545	343	2.46e-01	1.12e-01	8.27e-04	9.15e-03
11	756	512	3.90e-01	3.12e-01	1.56e-03	1.23e-02
12	1019	729	6.21e-01	7.70e-01	2.85e-03	1.59e-02
13	1340	1000	9.65e-01	1.63e+00	5.18e-03	2.13e-02
14	1725	1331	1.43e+00	3.25e+00	8.56e-03	2.73e-02
15	2180	1728	2.13e+00	6.29e+00	1.37e-02	3.52e-02
20	5715	4913	1.34e+01	1.01e+02	9.30e-02	9.49e-02

Fig. 3. Setting up the element stiffness matrix - computing time - 2D

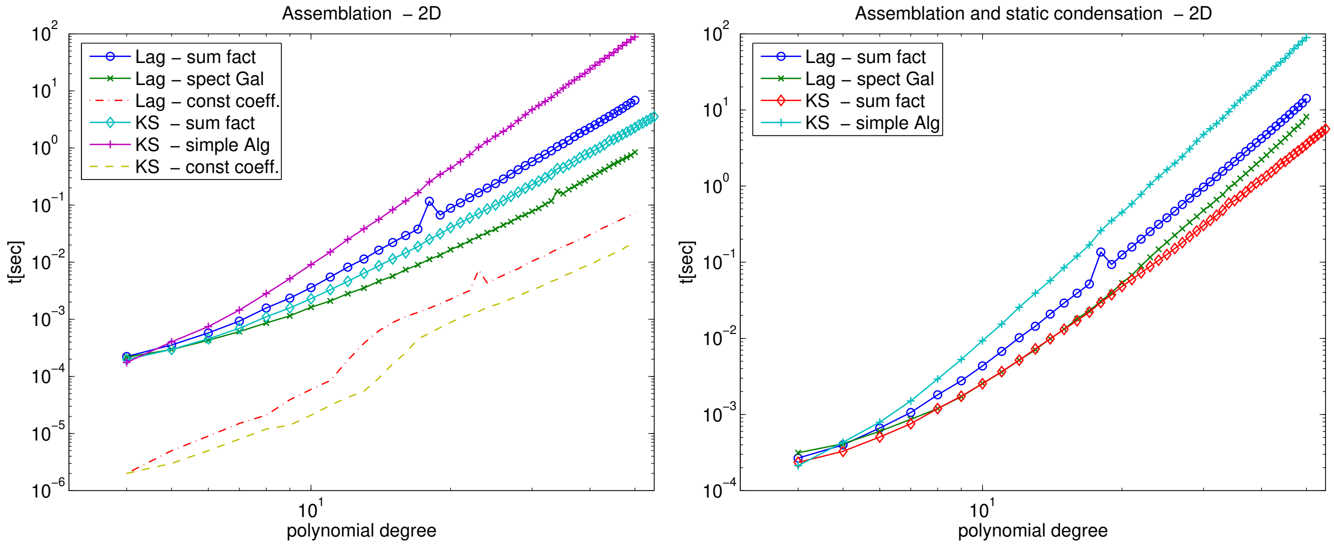


Fig. 4. Setting up the element stiffness matrix - computing time - blockwise - 2D

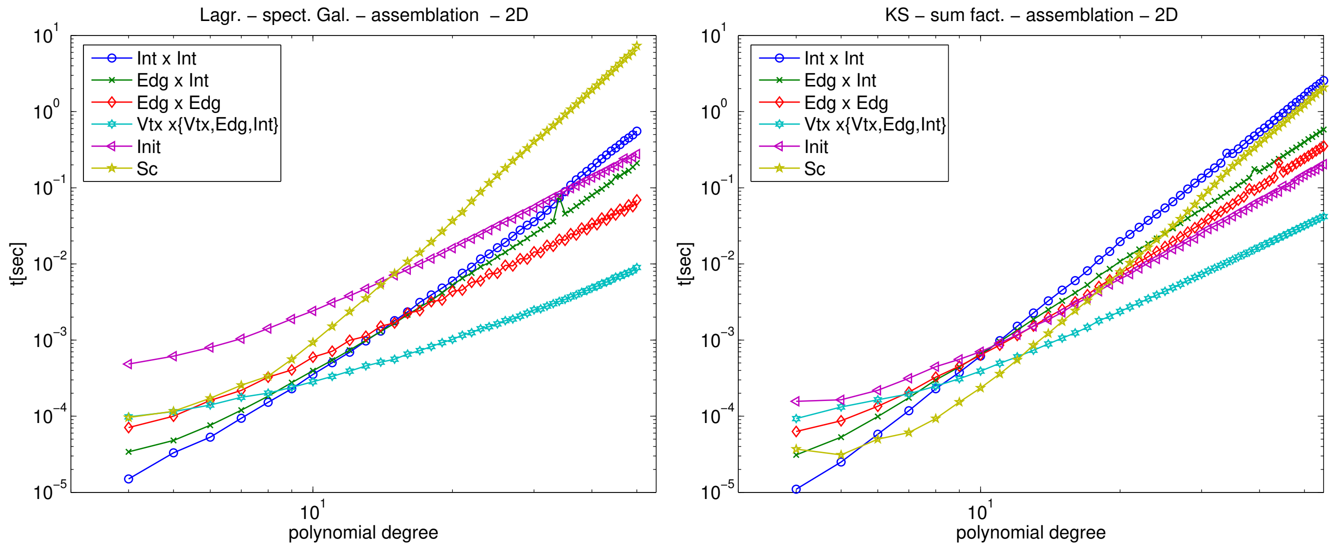


Fig. 5. Approximation properties (cf. Example 4.1)

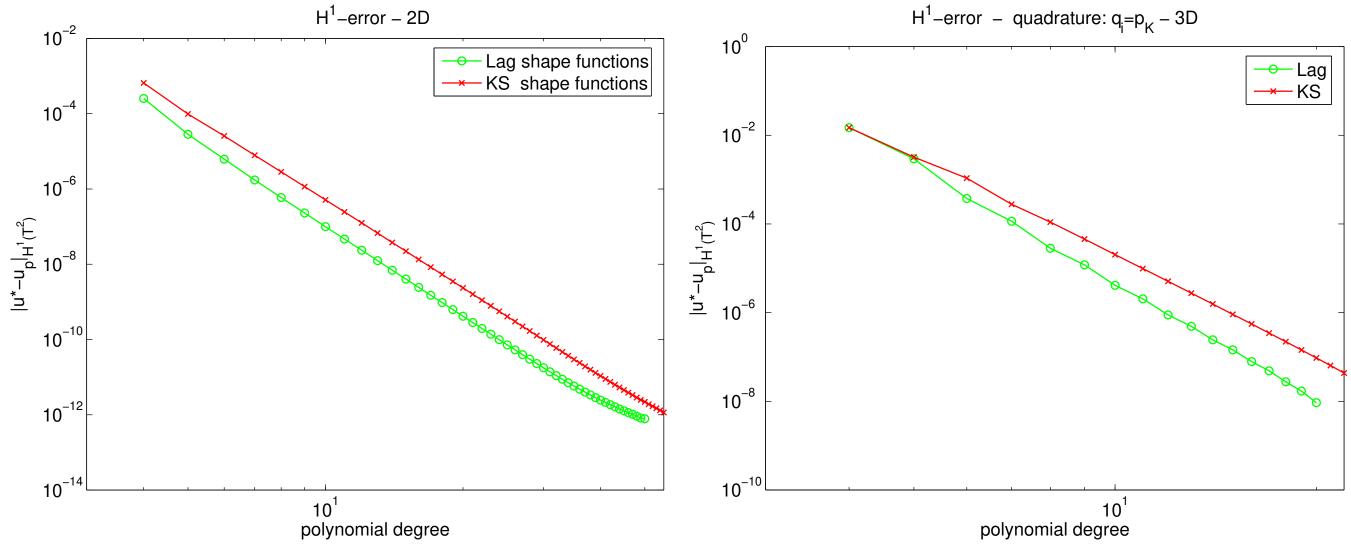


Fig. 6. Setting up the element stiffness matrix - computing time - 3D

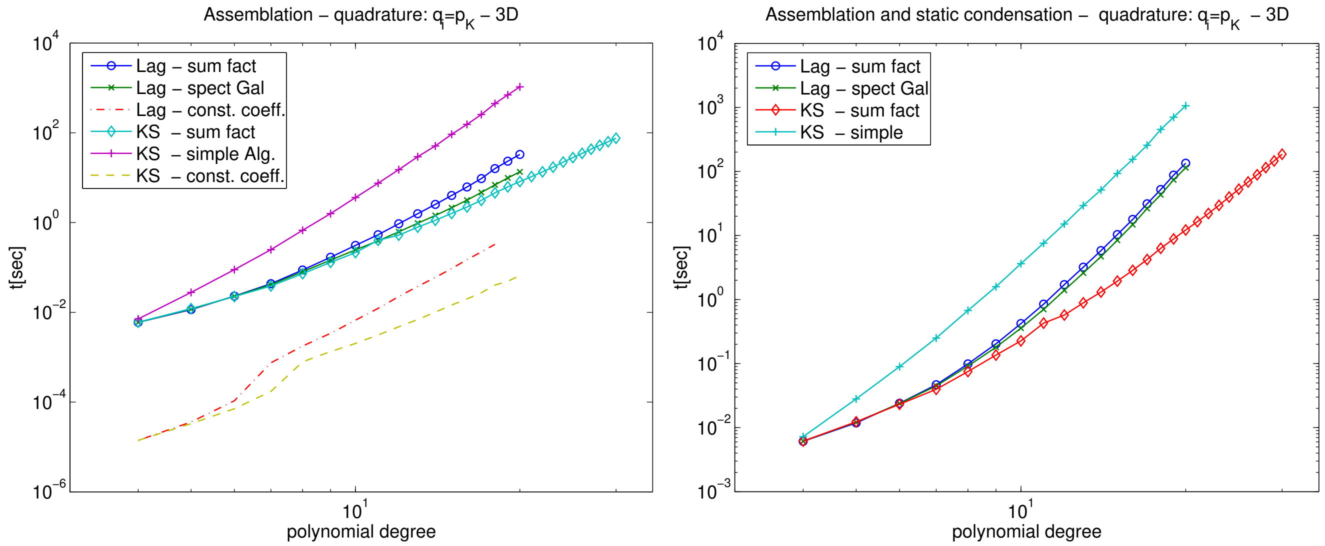


Fig. 7. Setting up the element stiffness matrix - computing time - blockwise - 3D

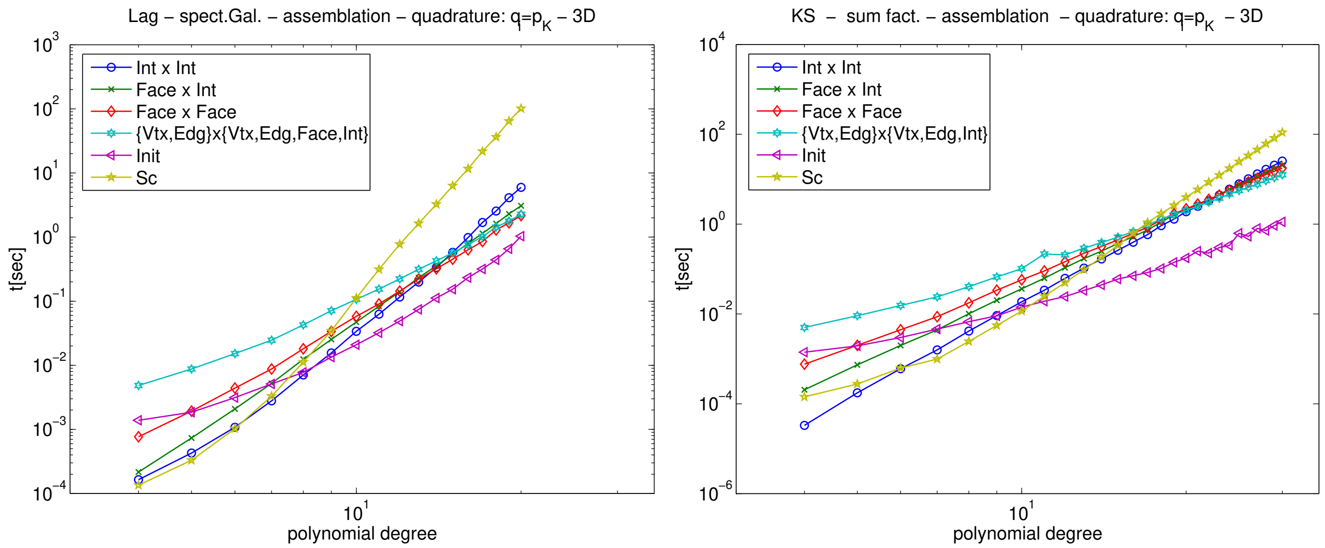


Fig. 8. Matrix vector multiplication - computing time

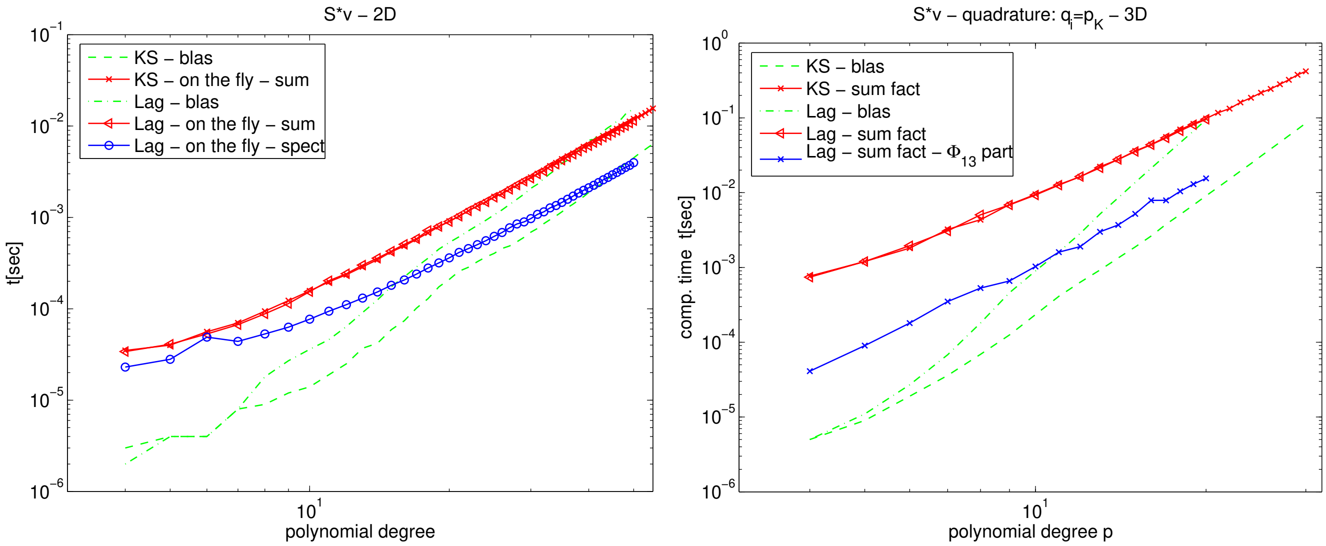
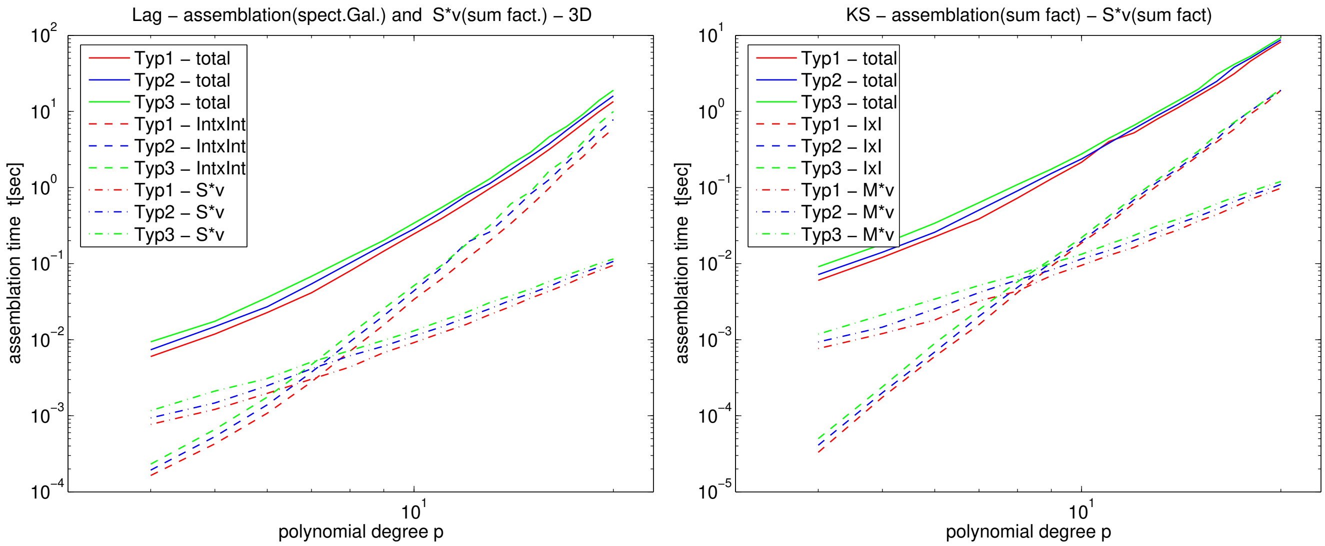


Fig. 9. Different quadrature rules - assemblation and matrix vector multiplication - computing time - 3D



IX. Conclusions

In the last sections we discussed different algorithms for setting up the element stiffness matrix and performing on the fly matrix vector multiplication. We considered the Karniadakis-Sherwin shape functions $\Phi^{(KS)}$ in combination with the standard algorithm and sum factorization and introduced modified shape functions Φ^{Lag} to generalized the spectral Galerkin ideas of [7] to triangular and tetrahedral elements. The numerical results of Section VIII show the following:

- The standard method is by far the slowest algorithm in any case.
- Due to the increased number of internal shape functions, using $\Phi^{(Lag)}$ leads to better approximation results. Moreover, for the two dimensional case setting up the element stiffness matrix with the spectral Galerkin algorithm in conjunction with $\Phi^{(Lag)}$ is significantly faster than setting up the stiffness matrix for $\Phi^{(KS)}$ with sum factorization. For 3 dimensions the shape functions $\Phi^{(Lag)}$ contain almost six times the number of internal shape functions as $\Phi^{(KS)}$ does and the computing time for setting up the element stiffness matrix with the spectral Galerkin algorithm in conjunction with $\Phi^{(Lag)}$ is within the range ($p_K \leq 20$) almost the same as setting up the stiffness matrix for $\Phi^{(KS)}$ with sum factorization.
- Due to the increased number of internal shape functions, the static condensation process for the element stiffness matrix based on $\Phi^{(Lag)}$ is considerable slower as the static condensation process for the element stiffness matrix based on $\Phi^{(KS)}$. However, for the 2-dimensional case and $p_K \leq 20$ the computing time for setting up the condensed element stiffness

matrix is in the case of $\Phi^{(Lag)}$ in conjunction with the spectral Galerkin algorithm almost the same as for the case of $\Phi^{(KS)}$ together with sum factorization. In the 3-dimensional case this is already reached at $p_K = 8$.
- A significant speed-up for setting up the element stiffness matrix can be obtained on elements with constant coefficients.
- Considering a hp-implemention based on the fly matrix vector multiplications, we obtain in the 2-dimensional case a significant speed-up using the modified shape functions $\Phi^{(Lag)}$ in conjunction with Algorithm 7.6.

REFERENCES

[1] C. Bernardi and Y. Maday. Spectral methods. In P.G. Ciarlet and J.L. Lions, editors, *Handbook of Numerical Analysis, Vol. 5*. North Holland, 1997.
[2] L. Demkowicz, K. Gerdes, C. Schwab, A. Bajer, and T. Walsh. A general and flexible fortran 90 hp-FE code. *Computing and Visualization in Science*, 1:145–163, 1998.
[3] L. Demkowicz, T.J. Oden, W. Rachowicz, and O. Hardy. Towards a universal hp finite element strategy. part 1. constrained approximation and data structure. *Comput. Meth. Appl. Mech. Engrg.*, 77:79–112, 1989.
[4] T. Eibner. *Algorithmik der randkonzentrierten FEM*. PhD thesis, TU Chemnitz, (in prep.).
[5] G.E. Karniadakis and S.J. Sherwin. *Spectral/hp Element Methods for CFD*. Oxford University Press, 1999.
[6] Y. Maday and E.M. Rønquist. Optimal error analysis of spectral methods with emphasis on non-constant coefficients and deformed geometries. *Comput. Meth. Appl. Mech. Engrg.*, 80:91–115, 1990.
[7] J.M. Melenk, K. Gerdes, and C. Schwab. Fully discrete hp-FEM: fast quadrature. *Comput. Meth. Appl. Mech. Engrg.*, 190:4339–4364, 2001.
[8] J.M. Melenk and C. Schwab. hp FEM for reaction-diffusion equations I: Robust exponentional convergence. *SIAM J. Numer. Anal.*, 35:1520–1557, 1998.
[9] J.M. Melenk and B. Wohlmuth. On residual-based a posteriori error estimation in hp-FEM. *Advances in Comp. Math.*, 15:311–331, 2001.
[10] S.A. Orszag. spectral methods for problems in complex geometries. *J. Comput. Phys.*, 37:70–92, 1980.
[11] C. Schwab. *p- and hp-Finite Element Methods*. Oxford University Press, 1998.
[12] B. Szabó and I. Babuška. *Finite Element Analysis*. Wiley, 1991.