Eidgenössische Technische Hochschule Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Fully Discrete *hp*-Finite Elements: Fast Quadrature

J.M. Melenk, K. Gerdes* and C. Schwab

Research Report No. 99-15
September 1999

Seminar für Angewandte Mathematik
Eidgenössische Technische Hochschule
CH-8092 Zürich
Switzerland

*Department of Mathematics, Chalmers University, SE-412 96 Göteborg, Sweden

Fully Discrete *hp*-Finite Elements: Fast Quadrature

J.M. Melenk, K. Gerdes* and C. Schwab

Seminar für Angewandte Mathematik
Eidgenössische Technische Hochschule
CH-8092 Zürich
Switzerland

Research Report No. 99-15 September 1999

Abstract

A fully discrete *hp* finite element method is presented. It combines the features of the standard *hp* finite element method (conforming Galerkin Formulation, variable order quadrature schemes, geometric meshes, static condensation) and of the spectral element method (special shape functions and spectral quadrature techniques). The speed-up (relative to standard *hp* elements) is analyzed in detail both theoretically and computationally.

Keywords: *hp* finite element method, spectral element method, numerical integration

Subject Classification: Primary: 65N30, 65N35 Secondary: 65N50

*Department of Mathematics, Chalmers University, SE-412 96 Göteborg, Sweden

1 Introduction

The Finite Element Method (FEM) is today the most widely used discretization technique in solid and fluid mechanics. The classical approach still dominant in today's industrial applications is to partition the domain into many small subdomains of diameter $O(h)$ and to approximate the unknown solution by a piecewise polynomial of low order p (typically in applications, $1 \leq p \leq 3$). Convergence is achieved through mesh refinement, i.e., by letting $h \to 0$.

In the early eighties, the so-called p-version FEM emerged where the polynomial degree $p \to \infty$ on a fixed mesh in order to produce convergent sequences of FE-solutions (see, e.g., [2, 24] and the references there). Very closely related to the p-version FEM are spectral methods and Spectral Element Methods developed at the same time (cf. [10, 17, 5, 18, 13, 14, 3] together with the references therein), which may be viewed as weighted collocation methods. It was noted early on that some of the Spectral Methods could be interpreted as p-version FEMs with certain types of quadrature. Being collocation methods, Spectral methods are much faster than the standard p-version FEM as the generation of the stiffness is considerably cheaper. Additionally, various fast transform techniques have been devised to speed up the generation of the stiffness matrix.

Many problems in fluid and solid mechanics have piecewise analytic solutions. For the approximation of such functions, the h-version, p-version FEM, and the Spectral methods can only lead to *algebraic* rates of convergence. In the context of the FEM, however, it is possible to combine the h and the p-version FEM into the hp-FEM where mesh refinement and an increase of the polynomial degree p can be done simultaneously. The proper combination of *geometric* mesh refinement towards the singularities and increase of the polynomial degree in regions where the solution is smooth (analytic) then leads to *exponential* rates of convergence (see [2, 19]). Commercial FEM codes with p and hp capabilities were developed since the mid 80ies. We mention here the pioneering code PROBE by B. Szabó and his co-workers for heat transfer and elasticity problems, now superseded by STRESSCHECK, [21], the code Stripe, [20], and PHLEX, [22]. A general 2-D hp-FEM code for various problems from solid and fluid mechanics was developed in the late 80ies by [6, 7]. There, general meshes consisting of triangles and/or quadrilaterals and even certain types of irregular ("hanging") nodes were admissible, which facilitate mesh refinement and allow the construction of geometrically refined meshes in a simple manner. More recently an improved implementation of this hp-FEM code has been presented in [8].

As mentioned, the key to the success of the hp-FEM is the ability to combine p refinement with mesh refinement. In contrast to this, the (original) Spectral method is based on a fixed partitioning of the computational domain into (curvilinear) rectangles (hyper cubes in more than 2 dimensions) and then increasing the spectral order. Among the spectral element community, the observed need for mesh refinement has lead to various non-conforming approaches such as the use of "mortar elements" (see, e.g., [4] and the references therein), which also exhibit hp-like features.

From an approximation theory point of view, the flexibility of the hp-FEM to combine h and p refinement makes it clearly superior to more traditional h-, p- version FEMs and Spectral methods. The popularity of h-FEM is due to simple stiffness matrix generation and available fast multilevel solvers (e.g., [11]), that of Spectral methods is due to the

availability of very efficient techniques for the stiffness matrix generation and preconditioners. In contrast, the perception of *hp*-FEMs is that they are computationally expensive in their stiffness matrix generation and that fast solution algorithms do not seem to exist, although domain decomposition methods [15], combined with parallelization, [16], can be very effective.

This paper is devoted to the development of *hp*-spectral Galerkin FEMs which unify and generalize "standard" *hp*-FEM technology and spectral methods. The most prominent features shared with *hp*-FEM are the decoupling of the quadrature rule from the elemental polynomial degrees and the ability to employ mesh refinement via hanging nodes. In the solution process, local static condensation and the element stiffness matrices are obtained completely in parallel. Geometry representation is uncoupled from the ansatz space, accomodating exact representations of the geometry via blending methods and the use of *hp*-isoparametric elements that approximate the geometry. Spectral techniques, are well-known to speed up the element stiffness matrix generation considerably. The main idea in spectral methods is combining Gauss-Lobatto quadrature with the use of shape functions that are Lagrange interpolation polynomials in the quadrature points. In spectral methods, the quadrature order is thus linked to the elemental polynomial degree and Gauss-Lobatto quadrature implies underintegration even for affine element maps in 2-D.

Here, we generalize the spectral quadrature to polynomial degrees p and quadrature rules of order $p+q$, $q \geq 0$ arbitrary. We show that the work per element is $O(p^4(1+q))$ and $O(qp^6+p^5)$ for problems in $\mathbb{R}^2$, $\mathbb{R}^3$ respectively (see Table 5 ahead for the details) [1]. This is to be compared with $O(p^{3d})$ for problems in $\mathbb{R}^d$ for the standard quadrature algorithm. We also show numerically that this speed-up is already visible for spectral orders p in the range of $5 \leq p \leq 10$.

We finally mention that similar algorithms have been proposed in, e.g., [12] for meshes based on triangles/tetrahedra; the *hp* convergence analysis of these algorithms will be the topic of a future paper.

The outline of our presentation is as follows: In Section 2, the spectral Galerkin Algorithm (Algorithm 2.12) is presented. For clarity of exposition, the algorithm is presented for a 2D scalar model problem. The extension to higher dimensions is discussed in Section 3.1. Section 3.2 demonstrates for a 3D hexahedral element that Algorithm 2.12 does indeed realize significant computational savings already at moderate polynomial degrees p. Some concluding remarks finish the paper.

2 *hp* Spectral Quadrature Algorithm

2.1 Model problem

We will illustrate the *hp*-spectral Galerkin algorithm for the following scalar model problem in two dimensions

$$-\nabla \cdot (A(x)\nabla u) = f(x) \qquad \text{on } \Omega \subset \mathbb{R}^2, \qquad u = 0 \qquad \text{on } \partial\Omega \tag{2.1}$$

[1] here and in what follows the implied constant in the $O(\cdot)$ notation is independent of both p and q

where the matrix A is symmetric, positive definite, and piecewise smooth. The weak form is: Find $u \in H_0^1(\Omega)$ such that

$$a(u,v) = l(v) \qquad \forall v \in H_0^1(\Omega)$$

where the bilinear form a and the right hand side l are given by

$$a(u,v) = \int_\Omega \nabla v \cdot (A(x)\nabla u)\, dx\, dy, \qquad l(v) = \int_\Omega f v\, dx\, dy.$$

We restrict here our attention to the model problem (2.1) for simplicity of notation. The *hp*-spectral Galerkin algorithm adapts straight forwardly for equations involving lower order terms and vector valued problems.

2.2 *hp*-FEM

In the *hp*-FEM the domain Ω is partitioned into elements K; each element K is the image of the master element $\hat{K} = (0,1)^2$ under the (bijective) element map

$$F_K : \hat{K} = (0,1)^2 \to K. \tag{2.2}$$

The elements K are collected into the triangulation $\mathcal{T} = \{K\}$. The *hp* FE-space is then a space of continuous, mapped polynomials:

$$S = \{u \in H_0^1(\Omega) \,|\, u|_K \circ F_K \in S_{p_K}(\hat{K})\} \tag{2.3}$$

where the spaces $S_{p_K}(\hat{K})$ are spaces of polynomials of degree p_K on the master element $\hat{K}$. The subscript K in p_K indicates that we allow for variable polynomial degree (see Section 2.4 for a detailed description of the spaces $S_{p_K}(\hat{K})$). The *hp*-FEM then reads: find $u \in S$ such that $a(u,v) = l(v)$ for all $v \in S$. The bilinear form a and the right hand side functional l can be written as a sum of element integrals and the element maps F_K provide a means to express all integrals over elements K as integrals of the master element $\hat{K}$: find $u \in S$ s.t.

$$\begin{aligned}
\sum_{K\in\mathcal{T}} a_K(u,v) &= \sum_{K\in\mathcal{T}} l_K(v) \qquad \forall v \in S \\
a_K(u,v) &= \int_{\hat{K}} \nabla\hat{u} \cdot (\hat{A}\nabla\hat{v}) d\xi_1 d\xi_2 = \int_{\hat{K}} \sum_{i,j=1}^{2} \frac{\partial\hat{u}}{\partial\xi_i} \hat{A}_{ij} \frac{\partial\hat{v}}{\partial\xi_j} d\xi_1 d\xi_2 \\
&=: \sum_{i,j=1}^{2} a_K^{ij}(u,v) \\
l_K(u,v) &= \int_{\hat{K}} \hat{f}\hat{v} d\xi_1 d\xi_2 \\
\hat{A} &= (F_K')^{-T} (A \circ F_K) (F_K')^{-1} J_K \qquad (2.4) \\
\hat{f} &= f \circ F_K J_K \qquad (2.5) \\
\hat{u} &= u \circ F_K, \qquad \hat{v} = v \circ F_K
\end{aligned}$$

with $J_K = \det F_K'$. If $\{\varphi_i\}_{i=1}^N$ is a basis of $S_{p_K}(\hat{K})$, then the

$$A_K := (a_K(\varphi_j, \varphi_i))_{i,j=1}^N, \qquad L_K := (l_K(\varphi_i))_{i=1}^N$$

are the element stiffness matrix A_K and the element load vector L_K. In the *assembly process*, these element stiffness matrices (and load vectors) are assembed into the global linear system which can then be solved. Especially in the context of iterative solvers, it may be advantageous to combine several elements into a *mesh patch* (also known as substructure or subdomain) and first subassemble the elements of the mesh patch before assembling the patches. The generic *hp* Galerkin FEM algorithm reads as follows:

Algorithm 2.1 (*hp* Galerkin FEM)

```
loop over all mesh patches
  within each mesh patch loop over all elements
    generate element stiffness matrix A_K (and element load vector L_K)
    (optional) condense inner elemental dof
    assemble elemental dof into mesh patch stiffness matrix
  endloop over elements in mesh patch
  (optional) condense inner mesh patch dof
  assemble mesh patch dof into the global stiffness matrix
endloop over all mesh patches
solve global system
backsolve for inner mesh patch and inner element dof (optional)
```

Several comments on Algorithm 2.1 are in order. The main amount of (alphanumerical) work arises in the generation of A_K and L_K, due to the numerical quadrature unless the F_K are affine in which case the A_K can be precomputed once and for all. In general, however, curved domains entail complicated element maps F_K and numerical integration is required. Additionally, the evaluation of the element maps F_K may be expensive and isoparametric or "quasi-regional" mappings, [1], are employed instead. For an error analysis of such approximate mappings we refer to [9].

The most elementary quadrature algorithm is

Algorithm 2.2 (standard Galerkin element quadrature algorithm)

```
initialize A_K = 0
for all quadrature points
     for all basis functions φ of S_{p_K}
         for all basis functions ψ of S_{p_K}
             add contribution of a_K(φ,ψ) at this quadrature point to A_K
         end
     end
end
```

It is easy to see that in two dimensions for a quadrature rule with $O(p^2)$ points and polynomials of degree p, Algorithm 2.2 requires $O(p^6)$ work. Fig. 1 shows the CPU time spent on various components of Algorithm 2.1 for the polynomial degree p ranging from

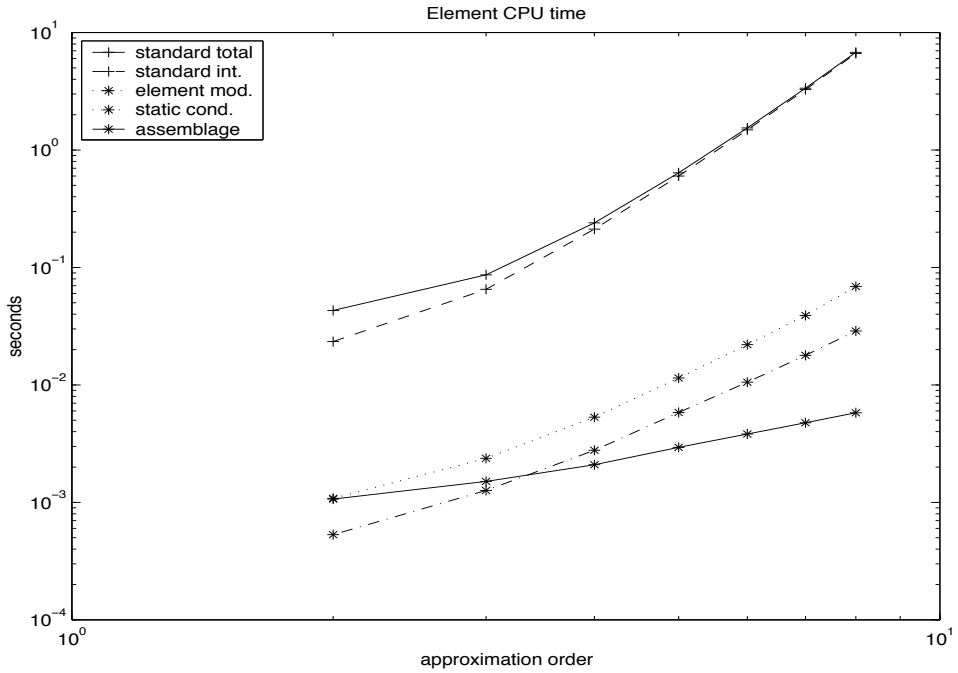

Figure 1: CPU time per quadrilateral element for standard element algorithms for scalar model problem.

$p = 1$ to $p = 8$ and the tensor product space Q_p, the space of polynomials of degree p in each variable:

$$Q_p(\hat{K}) = \text{span}\,\{x^i y^j \,|\, 0 \le i, j \le p\}, \qquad \dim Q_p(\hat{K}) = (p+1)^2. \tag{2.6}$$

We clearly see that indeed the majority of the work is spent on the numerical quadrature and not on other tasks such as condensation, enforcement of constraints, etc. Accordingly, the remainder of this paper is devoted to the development and analysis of various forms of accelerated quadrature routines. We propose to use Algorithm 2.12 below for the generation of A_K. One of the advantages of this algorithm is that it only affects the so-called internal degrees of freedom; this implies that Algorithm 2.12 could easily be incorporated in existing *hp*-FEM codes.

Remark 2.3 We formulated Algorithm 2.1 in the spirit of classical *hp*-FEM based on direct solvers by using the notion of static condensation, mesh patches, etc. We point out, however, that Algorithm 2.1 can also be employed in an environment where the global solution process is done with iterative techniques. Even a "completely iterative" scheme where condensation is not performed at all, is possible. Akin to spectral and spectral element methods, Algorithm 2.12 can be reformulated to realize a matrix vector multiplication rather than to generate explicitly the (local) stiffness matrices.

2.3 Element Stiffness Matrix Generation

The goal of the present subsection is to derive Algorithm 2.12, the Spectral Galerkin Element Algorithm, unifying the standard Galerkin FEM and the Spectral Element Method. We will only describe the construction of the element stiffness matrix A_K. The element

load vector L_K can be constructed using the same ideas but will have to account for possibly non-smooth data $f(x)$.

In order to derive Algorithm 2.12, we start by recalling Gauss-Lobatto quadrature in Subsection 2.3.1. Next, we describe a modified sum factorization technique (Algorithm 2.5). Algorithm 2.10, which is at the heart of Algorithm 2.12, can then be understood as a degenerate form of Algorithm 2.5. As mentioned above, we permit elements with general polynomial degree. However, in order to illustrate the complexity of the algorithms that we will describe below, we will always give an operation count for the case that $S_{p_K}(\hat{K}) = Q_p(\hat{K})$.

2.3.1 Gauss-Lobatto Quadrature

To evaluate the entries of A_K, one generally has to use quadrature rules. We will use tensor product Gauss-Lobatto quadrature rules. We recall that in one dimension, the $q+1$ Gauss-Lobatto points $\mathcal{GL}^q = \{x_0, \ldots, x_q\}$ are the zeros of the polynomial $x \mapsto x(1-x)L_q'(x)$ where L_q stands for q-th Legendre polynomial associated with the unit interval $(0,1)$. It is well-known [3], that these zeros are distinct and lie in the interval $[0,1]$. Furthermore, for each q one can find positive weights w_i, $i = 0, \ldots, q$ such that the quadrature rule

$$GL_q(u) := \sum_{k=0}^{q} w_k u(x_k) \approx \int_0^1 u(x)\, dx$$

is exact for polynomials u of degree $2q-1$, [3]. Gauss-Lobatto quadrature rules in two dimensions are then obtained by tensor product constructions. For example, quadrature rules $GL_{q_1,q_2}(u)$ based on the points $\mathcal{GL}^{q_1} \times \mathcal{GL}^{q_2}$ are given by

$$GL_{q_1,q_2}(u) := \sum_{k=0}^{q_1} \sum_{l=0}^{q_2} w_{1,k} w_{2,l} u(x_{1,k}, x_{2,l}) \tag{2.7}$$

For simplicity of notation in some of the ensuing algorithms, we assume that $x_{1,0} = 0$, $x_{1,q_1} = 1$, $x_{2,0} = 0$, $x_{1,q_2} = 1$.

Once a quadrature rule is chosen, the evaluation of the entries of the stiffness matrix A_K (and of the load vector) proceeds by replacing integrals over $\hat{K}$ by (2.7). For example, for φ, $\psi \in S_{p_K}(\hat{K})$ the evaluation of $a^{rs}(\varphi, \psi)$ is performed as

$$a^{rs}(\varphi, \psi) \approx GL_{q_1,q_2}\left(\frac{\partial}{\partial \xi_r}\varphi \cdot \frac{\partial}{\partial \xi_s}\psi \cdot \hat{A}_{rs} \right), \quad r, s \in \{1, 2\}, \tag{2.8}$$

where the functions $\hat{A}_{rs}$ are defined in (2.4).

2.3.2 Sum Factorization

Let us assume that a basis of the space $S_{p_K}(\hat{K})$ is given in the form $S_{p_K}(\hat{K}) = \mathrm{span}\{\mathcal{E}^0 \cup \mathcal{E}^1 \cup \mathcal{E}^2 \cup \mathcal{E}^3 \cup \mathcal{E}^4 \cup \mathcal{I}\}$, where the sets $\mathcal{E}^k$, $\mathcal{I}$ have the following tensor product structure

$$\mathcal{E}^k = \{\chi^k_{1,i}(\xi_1)\chi^k_{2,j}(\xi_2)\,|, 1 \le i \le I_k, \quad 1 \le j \le J_k\}, \qquad k = 0, \ldots, 4, \tag{2.9}$$

$$\mathcal{I} = \{\chi^{\mathcal{I}}_{1,i}(\xi_1)\chi^{\mathcal{I}}_{2,j}(\xi_2)\,|, 1 \le i \le I_{\mathcal{I}}, \quad 1 \le j \le J_{\mathcal{I}}\}. \tag{2.10}$$

This structure is motivated by the usual decomposition into *vertex shape functions*, *side shape functions*, and *internal shape functions*. In that case, the numbers I_k, J_k, $I_{\mathcal{I}}$, $J_{\mathcal{I}}$ are a measure for the polynomial degree associated with each of these entities. For example, for the space $Q_p(\hat{K})$, a standard choice of these numbers is (see Section 2.4 for a more specific example)

$$I_{\mathcal{I}} = J_{\mathcal{I}} = p-1, \qquad I_0 = J_0 = 2 \tag{2.11}$$

$$(I_2, J_2) = (I_4, I_4) = (p-1, 1) \quad \text{for the sides parallel to the } y\text{-axis} \tag{2.12}$$

$$(I_1, J_1) = (I_3, J_3) = (1, p-1) \quad \text{for the sides parallel to the } x\text{-axis} \tag{2.13}$$

The element stiffness matrix A_K has a 6×6 block structure, corresponding to the interaction of the 6 different types of shape functions introduced in (2.9), (2.10). An algorithm that exploits the tensor product structure of the basis functions and is based on sum factorization is the following:

Algorithm 2.4

initialize $A_K = 0$
for r,s=1:2
  for $B_1 \in \{\mathcal{E}^0, \mathcal{E}^1, \mathcal{E}^2, \mathcal{E}^3, \mathcal{E}^4, \mathcal{I}\}$
    for $B_2 \in \{\mathcal{E}^0, \mathcal{E}^1, \mathcal{E}^2, \mathcal{E}^3, \mathcal{E}^4, \mathcal{I}\}$
    add sum_fact$\left(\frac{\partial}{\partial \xi_r} B_1, \frac{\partial}{\partial \xi_s} B_2, \hat{A}_{rs}\right)$ to the block B_1-B_2 of A_K
    end
  end
end

Here, the operators $\frac{\partial}{\partial \xi_r}$, $\frac{\partial}{\partial \xi_s}$ are understood to act elementwise on the shape function blocks B_1, B_2. Each call to sum_fact returns a $\dim B_1 \times \dim B_2$ matrix whose entries are just those calculated by the right hand side of (2.8) for all $\varphi \in B_1$, $\psi \in B_2$. We now specify the function sum_fact in Algorithm 2.4:

Algorithm 2.5 (sum factorization)

A := sum_fact(B_1, B_2, a)
  %input:
  % $B_1 = \{\varphi_i^1(\xi_1)\varphi_j^2(\xi_2) \,|\, 1 \le i \le I, \quad 1 \le j \le J\}$
  % $B_2 = \{\psi_i^1(\xi_1)\psi_j^2(\xi_2) \,|\, 1 \le i \le I', \quad 1 \le j \le J'\}$
  % function $a : \hat{K} \to \mathbb{R}$
  %output: the matrix A with entries
  %$A_{(i,j),(i',j')} = \sum_{k=0}^{q_1} \sum_{l=0}^{q_2} \varphi_i^1(x_{1,k})\varphi_j^2(x_{2,l})\psi_{i'}^1(x_{1,k})\psi_{j'}^2(x_{2,l})w_{1,k}w_{2,l}a(x_{1,k}, x_{2,l})$

$S_1 := (1+q_1)(1+q_2)J \cdot J' + (1+q_1)I \cdot I' \cdot J \cdot J'$ % flop count for $\sum_k \sum_l$
$S_2 := (1+q_1)(1+q_2)I \cdot I' + (1+q_2)I \cdot I' \cdot J \cdot J'$ % flop count for $\sum_l \sum_k$
if $S_1 \le S_2$ then { % choose summation order to minimize flop count
  compute auxiliary array $H(k, j, j') := \sum_{l=0}^{q_2} \varphi_j^2(x_{2,l})\psi_{j'}^2(x_{2,l})w_{2,l}a(x_{1,k}, x_{2,l})$,
  $k \in \{0, \ldots, q_1\}$, $j \in \{1, \ldots, J\}$, $j' \in \{1, \ldots, J'\}$
  compute entries $A_{(i,j),(i',j')} := \sum_{k=0}^{q_1} \varphi_i^1(x_{1,k})\psi_{i'}^1(x_{1,k})w_{1,k}H(k, j, j')$,
  $i \in \{1, \ldots, I\}$, $j \in \{1, \ldots, J\}$, $i' \in \{1, \ldots, I'\}$, $j' \in \{1, \ldots, J'\}$
}

```
else {
      compute auxiliary array H(l,i,i') := Σ_{k=0}^{q1} φ_j^1(x_{1,k}) ψ_{j'}^1(x_{1,k}) w_{1,k} a(x_{1,k}, x_{2,l}),
      l ∈ {0,...,q2}, i ∈ {1,...,I}, i' ∈ {1,...,I'}
      compute entries A_{(i,j),(i',j')} := Σ_{l=0}^{q2} φ_j^2(x_{2,l}) ψ_{j'}^2(x_{2,l}) w_{2,l} H(l,i,i'),
      i ∈ {1,...,I}, j ∈ {1,...,J}, i' ∈ {1,...,I'}, j' ∈ {1,...,J'}
}
return A
```

Remark 2.6 Algorithm 2.5 is essentially the classical sum factorization algorithm, e.g., [17], which is used in several commercial 3-D *hp*-FEM code. The new feature here over classical sum factorization is the comparison of S_1 and S_2. S_1 and S_2 estimate the operation count for the evaluation of the double sum as either $\sum_k \sum_l$ or as $\sum_l \sum_k$, and the smaller operation count is chosen. This modification of the classical sum factorization algorithm is particularly suited for elements S_{p_K} with highly anisotropic polynomial degree distribution and/or anisotropic quadrature rules.

Remark 2.7 We note in passing that another potential source of further savings is that the quadrature rule could be chosen for each pair B_1-B_2 separately. To illustrate this point, note that a typical choice for the vertex set $\mathcal{E}^0$ is the set of the 4 bilinear functions only and that the set of internal shape functions $\mathcal{I}$ consists of polynomials of degree p. Assuming for the moment that $a \equiv 1$ in Algorithm 2.5 we see that the rule $GL_{\lceil p/2 \rceil+1, \lceil p/2 \rceil+1}$ is sufficient to calculate $\texttt{sum\_fact}(\mathcal{E}^0, \mathcal{I}, \texttt{a})$ whereas $p+1$ points in each direction are required for the calculation in $\texttt{sum\_fact}(\mathcal{I}, \mathcal{I}, \texttt{a})$.

Example 2.8 (work for Algorithm 2.5) For the simplified case $q_1 = q_2 = p+q$, $q \geq 0$, inspection of Algorithm 2.5 shows that the complexity is

$$W = (1+p+q)^2 \min\{I \cdot I', J \cdot J'\} + (1+p+q) I \cdot I' \cdot J \cdot J'. \tag{2.14}$$

For our 2D model case $S_{p_K}(\hat{K}) = Q_p(\hat{K})$, we can be more explicit. Assuming that the sets $\mathcal{I}$ and $\mathcal{E}^k$ satisfy (2.11)–(2.13), (2.14) reveals that, for example, the call $\texttt{sum\_fact}(\frac{\partial}{\partial \xi_1}\mathcal{I}, \frac{\partial}{\partial \xi_2}\mathcal{I}, \hat{A}_{12})$ takes work $W = O(p^4(p+q) + p^2(p+q)^2)$. Similarly, the calls $\texttt{sum\_fact}(\frac{\partial}{\partial \xi_1}\mathcal{I}, \frac{\partial}{\partial \xi_2}\mathcal{E}^1, \hat{A}_{12})$ and $\texttt{sum\_fact}(\frac{\partial}{\partial \xi_1}\mathcal{E}^1, \frac{\partial}{\partial \xi_2}\mathcal{E}^2, \hat{A}_{12})$ are of complexity $W = O(p^3(p+q) + p(p+q)^2)$ and $W = O(p(p+q)^2 + p^2(p+q))$, respectively.

Remark 2.9 Clearly, Algorithm 2.5 can also be formulated for other tensor product quadrature rule, e.g., the tensor product Gauss-Legendre quadrature rule.

2.3.3 Spectral Galerkin Sum Factorization

Algorithm 2.5 only assumes that the shape functions and the quadrature scheme have tensor product structure. If, however, the shape functions and the quadrature scheme are adapted to each other, then further savings are possible. The classical spectral method may serve as an example (which we will elaborate below) where the shape functions are just the base polynomials for the Lagrange interpolation in the quadrature points. The following variant of Algorithm 2.5 allows us to exploit the resulting degeneracy.

Algorithm 2.10 (spectral Galerkin sum factorization)
A := spec_sum_fact(B_1, B_2, a)
%input:
% $B_1 = \{\varphi_i^1(\xi_1)\varphi_j^2(\xi_2) \,|\, 1 \le i \le I, \quad 1 \le j \le J\}$
% $B_2 = \{\psi_i^1(\xi_1)\psi_j^2(\xi_2) \,|\, 1 \le i \le I', \quad 1 \le j \le J'\}$
% function $a : \hat{K} \to \mathbb{R}$
%output: the matrix A with entries
%$A_{(i,j),(i',j')} = \sum_{k=0}^{q_1}\sum_{l=0}^{q_2} \varphi_i^1(x_{1,k})\varphi_j^2(x_{2,l})\psi_{i'}^1(x_{1,k})\psi_{j'}^2(x_{2,l})w_{1,k}w_{2,l}a(x_{1,k},x_{2,l})$

for each pair i, i' compute set $K(i,i') := \{k \in \{0,\dots,q_1\} \,|\, \varphi_i^1(x_{1,k})\psi_{i'}^1(x_{1,k}) \neq 0\}$
for each pair j, j' compute set $L(j,j') := \{l \in \{0,\dots,q_2\} \,|\, \varphi_j^2(x_{2,l})\psi_{j'}^2(x_{2,l}) \neq 0\}$
$S_1 := (1+q_1)\sum_{j,j'} |\, L(j,j') \,| + J \cdot J' \cdot \sum_{i,i'} |\, K(i,i') \,|$ % flop count for $\sum_k^{q_1}\sum_l^{q_2}$
$S_2 := (1+q_2)\sum_{i,i'} |\, K(i,i') \,| + I \cdot I' \cdot \sum_{j,j'} |\, L(j,j') \,|$ % flop count for $\sum_l^{q_2}\sum_k^{q_1}$
if $S_1 \le S_2$ then { % choose summation order to minimize flop count
compute auxiliary array $H(k,j,j') := \sum_{l \in L(j,j')} \varphi_j^2(x_{2,l})\psi_{j'}^2(x_{2,l})w_{2,l}a(x_{1,k},x_{2,l})$,
compute entries $A_{(i,j),(i',j')} := \sum_{k \in K(i,i')} \varphi_i^1(x_{1,k})\psi_{i'}^1(x_{1,k})w_{1,k}H(k,j,j')$,
}
else {
compute auxiliary array $H(l,i,i') := \sum_{k \in K(i,i')} \varphi_j^1(x_{1,k})\psi_{j'}^1(x_{1,k})w_{1,k}a(x_{1,k},x_{2,l})$,
compute entries $A_{(i,j),(i',j')} := \sum_{l \in L(j,j')} \varphi_j^2(x_{2,l})\psi_{j'}^2(x_{2,l})w_{2,l}H(l,i,i')$,
}
return A

Here, $|\, L(j,j') \,|$ and $|\, K(i,i') \,|$ stand for the number of elements of the sets $L(j,j')$, $K(i,i')$. We note that Algorithm 2.10 does indeed reduce to Algorithm 2.5 if the shape functions are not related to the quadrature rule: In that case $L(j,j') = \{0,\dots,q_2\}$, $K(i,i') = \{0,\dots,q_1\}$, and hence Algorithm 2.10 indeed coincides with Algorithm 2.5. Let us now show that Algorithm 2.10 does indeed lead to savings if the shape functions are adapted to the quadrature rule.

Example 2.11 (spectral method) The classical *spectral method* (i.e., the use of $GL_{p,p}$ in conjunction with the space $Q_p(\hat{K})$) is a special case of Algorithm 2.10. The shape functions $\mathcal{E}^k$ and $\mathcal{I}$ are assumed to satisfy (2.9), (2.10). The internal shape functions $\mathcal{I}$ are additionally assumed to be of the form

$$\chi_{1,i}^{\mathcal{I}}(x) = l_i^{(p)}(x, \mathcal{N}^p), \qquad i = 1,\dots,I_{\mathcal{I}} := p-1,$$
$$\chi_{2,j}^{\mathcal{I}}(x) = l_j^{(p)}(x, \mathcal{N}^p), \qquad j = 1,\dots,J_{\mathcal{I}} := p-1.$$

Here, the set $\mathcal{N}^p = \mathcal{GL}^p = \{x_i \,|, i = 0,\dots,p\}$ is the set of Gauss-Lobatto nodes and the polynomials $l_i^{(p)}(x, \mathcal{N}^p)$, $i = 0,\dots,p$ are the Lagrange interpolation polynomials of degree p associated with the set $\mathcal{N}^p$ and they are given by

$$l_j^{(p)}(x, \mathcal{N}^p) := \prod_{\substack{i=0 \\ i \neq j}}^{p} \frac{x - x_i}{x_j - x_i}. \tag{2.15}$$

We note that there holds

$$l_i^{(p)}(x_j, \mathcal{N}^p) = \delta_{ij}, \qquad i,j = 0,\dots,p.$$

Let us compute the cost of the call to `spec_sum_fact`$(\frac{\partial}{\partial\xi_1}\mathcal{I}, \frac{\partial}{\partial\xi_2}\mathcal{E}^1, \hat{A}_{12})$. In doing so, we do not assume that the shape functions comprising $\mathcal{E}^1$ are related to the quadrature rule. The sets $K(i,i')$, $L(j,j')$ are then seen to be

$$K(i,i') = \{0,\ldots,p\}, \qquad L(j,j') = \{j\}.$$

We now readily compute that this function call costs $W = O(p^4)$. For the call to the function `spec_sum_fact`$(\frac{\partial}{\partial\xi_1}\mathcal{I}, \frac{\partial}{\partial\xi_2}\mathcal{I}, \hat{A}_{12})$ we calculate

$$K(i,i') = \{i'\}, \qquad L(j,j') = \{j\}.$$

Hence, the work estimate also reduces to $W = O(p^4)$. The reader will convince himself that for any r, $s \in \{1,2\}$, $k \in \{0,\ldots,4\}$, the work estimate for the calls to the functions `spec_sum_fact`$(\frac{\partial}{\partial\xi_r}\mathcal{I}, \frac{\partial}{\partial\xi_s}\mathcal{I}, \hat{A}_{rs})$ and `spec_sum_fact`$(\frac{\partial}{\partial\xi_r}\mathcal{I}, \frac{\partial}{\partial\xi_s}\mathcal{E}^k, \hat{A}_{rs})$, is always $O(p^4)$.

Example 2.11 is in essence the classical spectral method (there, however, the external shape functions $\mathcal{E}^k$ are also related to the quadrature rule thus allowing for improving the constant in the work estimate $O(p^4)$). Note that the use of the Gauss-Lobatto rule $GL_{p,p}$ leads to *underintegration* as even for affine elements, the stiffness matrix is not computed exactly. For our scalar problem, we will show in [9] that the rule $GL_{p,p}$ in conjunction with the space $Q_p(\hat{K})$ leads to a stable method and gives, in polygons and for piecewise analytic solution, exponential rates of convergence. However, this "minimal" spectral quadrature scheme performs poorly in the presence of very distorted meshes. For vector-valued problems such as the Lamé equations, to the knowledge of the authors, stability of the Spectral Element Method (i.e., the use of the rule $GL_{p,p}$) is an open problem on meshes containing non-affine elements. These considerations make it desirable to allow for *overintegration*, i.e., the ability to use the rule $GL_{p+q,p+q}$ ($q \geq 0$) in conjunction with the space $Q_p(\hat{K})$. We will show in the ensuing subsection that this is possible with work $W = O(p^4(1+q) + p^2q^2)$.

2.3.4 Transition from Spectral to Galerkin via overintegration

Following Example 2.11, we restrict our attention first to the quadrature rules $GL_{p+q,p+q}$ in conjunction with the space $Q_p(\hat{K})$. The external shape functions $\mathcal{E}^k$ are again chosen to satisfy (2.9). For a set of nodal points $\mathcal{N}^p$ satisfying

$$\{0,1\} \subset \mathcal{N}^p \qquad |\,\mathcal{N}^p\,| = p+1, \tag{2.16}$$

the internal shape functions $\mathcal{I}$ are Lagrange interpolation polynomials for this nodal set $\mathcal{N}^p$ (cf. (2.15)) and given by

$$\chi^{\mathcal{I}}_{1,i}(x) = l^{(p)}_i(x,\mathcal{N}^p), \qquad i = 1,\ldots,I_{\mathcal{I}} := p-1,$$
$$\chi^{\mathcal{I}}_{2,j}(x) = l^{(p)}_j(x,\mathcal{N}^p), \qquad j = 1,\ldots,J_{\mathcal{J}} := p-1.$$

It remains to choose the nodal set $\mathcal{N}^p$. Let $\mathcal{GL}^{p+q} = \{x_i \,|\, i = 0,\ldots,p+q\}$ be the Gauss-Lobatto quadrature points for the rule GL_{p+q} and $\mathcal{N}^p$ be any subset of $\mathcal{GL}^{p+q}$ satisfying (2.16). For such a choice of $\mathcal{N}^p$ there holds

$$\Big|\,\{x_j \in \mathcal{GL}^{p+q} \,|\, l^{(p)}_i(x_j,\mathcal{N}^p) \neq 0\}\,\Big| = \big|\,(\{i\} \cup \mathcal{GL}^{p+q} \setminus \mathcal{N}^p)\,\big| = 1+q, \qquad i = 1,\ldots,p-1.$$

In this situation, let us calculate the cost of evaluating `spec_sum_fact`$(\frac{\partial}{\partial\xi_1}\mathcal{I}, \frac{\partial}{\partial\xi_2}\mathcal{E}^2, \hat{A}_{12})$. We see that

$$\begin{aligned} |\,K(i,i')\,| &= \left|\,\mathcal{GL}^{p+q}\,\right| = |\,\{0,\ldots,p+q\}\,| = p+q+1, \\ |\,L(j,j')\,| &= \left|\,\left(\{j\}\cup(\mathcal{GL}^{p+q}\setminus\mathcal{N}^p)\right)\,\right| = 1+q. \end{aligned}$$

Inspection of Algorithm 2.10 shows that the work estimate is then $W = O((p+q)p(p^2+q))$. Similarly, for the call of `spec_sum_fact`$(\frac{\partial}{\partial\xi_1}\mathcal{I}, \frac{\partial}{\partial\xi_2}\mathcal{I}, \hat{A}_{12})$ we get

$$\begin{aligned} |\,K(i,i')\,| &= \left|\,\left(\{i'\}\cup\mathcal{GL}^{p+q}\setminus\mathcal{N}^p\right)\,\right| = q+1, \\ |\,L(j,j')\,| &= \left|\,\left(\{j\}\cup(\mathcal{GL}^{p+q}\setminus\mathcal{N}^p)\right)\,\right| = q+1. \end{aligned}$$

Thus, the work can be bounded by $W = O(p^2(p+q)(1+q))+O(p^4(q+1)) = O(p^4(1+q)+p^2q^2))$. In two dimensions, the use of the Gauss-Lobatto rule of order $p+q$ with a fixed $q \geq 0$ does therefore have the same asymptotic complexity as the pure spectral method of Example 2.11. We presented these ideas here for the case of the rule $GL_{p+q,p+q}$ for the space $Q_p(\hat{K})$. Similar arguments apply for more general polynomial degree distribution and anisotropic quadrature rules.

2.3.5 Choice of the interpolation nodes $\mathcal{N}^p$

In the preceding subsection 2.3.4, we merely assumed that the interpolation nodes $\mathcal{N}^p$ were a subset of the quadrature points $\mathcal{GL}^{p+q}$. In the present subsection, we discuss criteria for the selection of the interpolation set $\mathcal{N}^p$ and suggest some specific choices.

For a given nodal set $\mathcal{N}^p$ satisfying (2.16) we consider two types of 1-D shape functions on $(0,1)$:

$$\chi^1_0(x) := 1-x, \quad \chi^1_p(x) := x, \quad \chi^1_i(x) := l^{(p)}_i(x,\mathcal{N}^p), \quad i = 1,\ldots,p-1, \tag{2.17}$$

$$\chi^2_i(x) := l^{(p)}_i(x,\mathcal{N}^p), \quad i = 0,\ldots,p. \tag{2.18}$$

The set χ^1_i is motivated by the traditional *hp*-FEM codes that always include the linear (bilinear/trilinear shape functions in 2D/3D) shape functions. The set χ^2_i is closer to the typical choices in spectral methods. Note that these 1-D shape functions were used in the preceding subsection 2.3.4 to define the internal shape functions–this is the main motivation for imposing (2.16).

For these two sets of shape functions (2.17), (2.18), we can define the *mass matrix* M^m ($m \in \{1,2\}$) in the usual fashion:

$$M^m_{ij}(\mathcal{N}^p) := \int_0^1 \chi^m_i(x)\chi^m_j(x)\,dx, \qquad i,j = 0,\ldots,p+1.$$

The two mass matrices $M^m(\mathcal{N}^p)$, $m \in \{1,2\}$ depend of course on the nodal set $\mathcal{N}^p$ through the shape functions χ^m_i, and we included the argument $\mathcal{N}^p$ in the definition of M^m in order to emphasize this dependence. Given $p \geq 1$, $q \geq 0$, the first criterion for the selection of the subset $\mathcal{N}^p$ of $\mathcal{GL}^{p+q}$ is that the mass matrix M^m ($m \in \{1,2\}$) have minimal condition number. The corresponding *optimal* choice of the nodal set is then denoted by $\mathcal{N}^{p,q}_1$ for the shape functions (2.17) and $\mathcal{N}^{p,q}_2$ for the shape functions (2.18):

$$\mathcal{N}^{p,q}_m := \arg\min\{\mathrm{cond}\,(M^m(\mathcal{N}^p))\;|\,\mathcal{N}^p \subset \mathcal{GL}^{p+q} \text{ and } \mathcal{N}^p \text{ satisfies (2.16)}\}, \quad m \in \{1,2\}. \tag{2.19}$$

Tables 1, 2 give the optimal sets $\mathcal{N}_1^{p,q}$, $\mathcal{N}_2^{p,q}$ (up to symmetry); in the interest of brevity, Tables 1, 2 list the indices of the sets $\mathcal{GL}_{p+q} \setminus \mathcal{N}_m^{p,q}$ ($m \in \{1,2\}$) rather than the indices of the sets $\mathcal{N}_m^{p,q}$. Figs. 2, 3 show the condition numbers for these optimal choices of nodes. It is noteworthy that especially in the case of the shape functions (2.17), the condition number is fairly insensitive to q, that is, it is possible to simultaneously use overintegration and adapt the shape functions to the quadrature rule without an adverse effect on the condition number.

Another measure of the quality of a set of nodal points $\mathcal{N}^p$ is its Lebesgue number $\Lambda(\mathcal{N}^p)$, which is the stability constant of the corresponding nodal interpolation operator:

$$\Lambda(\mathcal{N}^p) := \sup_{x\in(0,1)} \sum_{i=0}^{p} \left| l_i^{(p)}(x) \right|. \tag{2.20}$$

Figs. 4, 5 show the Lebesgue numbers for our optimal sets $\mathcal{N}_m^{p,q}$, $m \in \{1,2\}$ (the Lebesgue numbers were calculated numerically by replacing the interval $(0,1)$ in (2.20) with 2000 uniformly distributed points). The Gauss-Lobatto set $\mathcal{GL}^p$ (corresponding to $q = 0$ in Figs. 4, 5), is essentially the best possible choice with $\Lambda(\mathcal{GL}^p) = O(\ln p)$, [23]. Nevertheless, in the practical regime $p = 2, \ldots, 20$ the Lebesgue numbers $\Lambda(\mathcal{N}_m^{p,q})$ are at most one order of magnitude away from the "optimal" value $\Lambda(\mathcal{GL}^p)$.

We note that in both Tables 1, 2, the optimal points are not distributed symmetrically with respect to the mid-point 1/2. For implementational purposes during the assembly procedure, it is more convenient to have symmetrically distributed nodal points $\mathcal{N}^p$ as this leads to shape functions that are either symmetric or anti-symmetric. This motivates to restrict the minimization in (2.19) to sets $\mathcal{N}^p$ that are symmetric with respect to the midpoint 1/2:

$$\begin{aligned}\mathcal{N}_{m,sym}^{p,q} := \arg\min \{ &\mathrm{cond}\,(M^m(\mathcal{N}^p)) \mid \\ &\mathcal{N}^p \subset \mathcal{GL}^{p+q},\ \mathcal{N}^p \text{ satisfies (2.16)},\ \mathcal{N}^p \text{ is sym. w.r.t. } 1/2 \},\quad m \in \{1,2\}.\end{aligned}$$

The optimal sets are listed in Tables 3, 4 (again, Tables 3, 4 actually list the indices of $\mathcal{GL}_{p+q} \setminus \mathcal{N}_{m,sym}^{p,q}$). Figs. 6, 7 show the corresponding condition numbers. Comparing these with Figs. 2, 3 we note that imposing a symmetry condition on the nodal sets $\mathcal{N}^p$ has practially no effect on the conditioning of the resulting mass matrix. A similar conclusion holds for the Lebesgue numbers as can be seen by comparing Figs. 8, 9 and Figs. 4, 5.

p	2	3	4	5	6	7	8	9	10
q=0	-	-	-	-	-	-	-	-	-
q=1	1	3	3	3	3	4	4	6	6
q=2	1,3	1,4	2,4	2,5	1,6	2,7	2,9	2,9	1,10
q=3	1,3,4	1,3,5	2,4,6	2,4,6	2,4,7	2,5,8	2,5,9	2,6,10	2,7,11
q=4	1,2,4,5	1,2,4,6	1,3,5,7	2,4,6,8	2,4,6,8	2,4,7,9	2,5,7,10	2,5,8,11	2,5,10,11
q=5	1,2,4,5,6	1,2,4,6,7	1,2,4,6,8	1,3,5,7,9	2,4,6,8,10	2,4,6,8,10	2,4,7,9,11	2,4,7,9,12	2,5,7,10,13
q=6	1,2,3,5,6,7	1,2,3,5,6,8	1,3,4,6,7,9	1,3,5,7,9,10	2,3,5,7,9,10	2,3,6,8,10,11	2,4,6,8,10,12	2,3,6,9,12,13	2,5,7,9,11,14

Table 1: index lists of $\mathcal{GL}^{p+q} \setminus \mathcal{N}_1^{p,q}$.

p	2	3	4	5	6	7	8	9	10
q=0	-	-	-	-	-	-	-	-	-
q=1	2	2	2	3	3	4	4	5	6
q=2	1,3	1,4	2,5	2,5	2,6	2,7	3,7	3,8	3,9
q=3	1,2,4	1,3,5	1,3,6	1,4,7	2,5,8	2,5,8	1,5,9	2,6,10	2,6,10
q=4	1,2,4,5	1,3,5,6	1,3,5,7	1,3,5,8	1,3,6,9	1,4,7,10	1,4,7,10	2,5,8,11	1,5,9,13
q=5	1,2,3,5,6	1,2,4,6,7	1,3,5,7,8	1,3,5,7,9	1,3,5,7,10	1,3,6,9,11	1,3,6,9,12	1,4,7,10,13	2,5,8,11,14
q=6	1,2,3,5,6,7	1,2,4,5,7,8	1,2,4,6,8,9	1,2,4,6,8,10	1,3,5,7,9,11	1,3,5,7,9,12	1,3,6,8,11,13	1,3,6,9,12,14	1,3,6,9,12,15

Table 2: index lists of $\mathcal{GL}^{p+q} \setminus \mathcal{N}_2^{p,q}$.

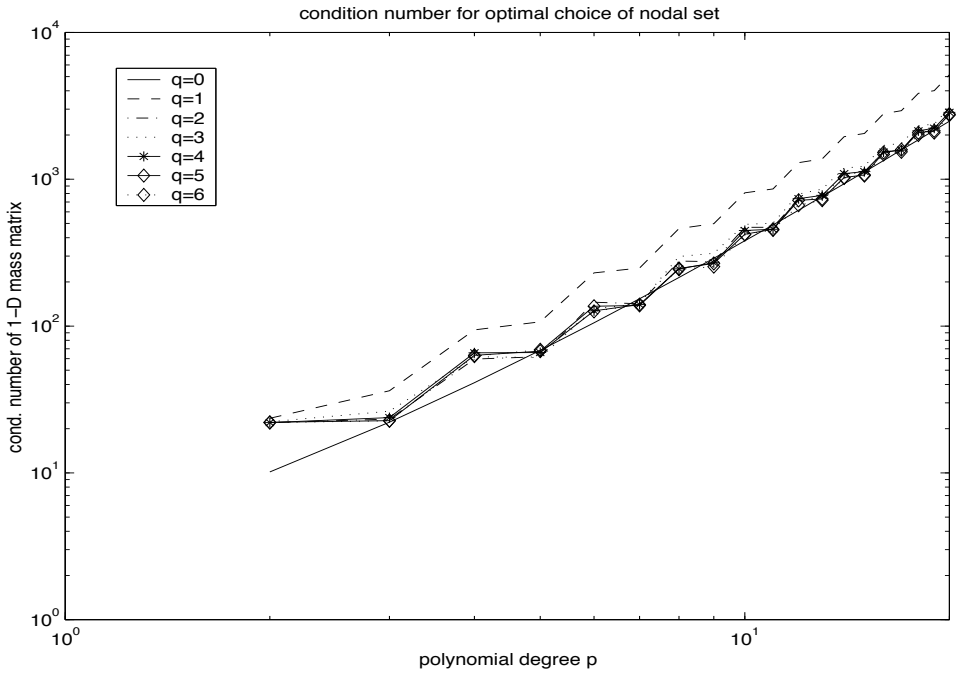

Figure 2: $\mathrm{cond}\,(M^1(\mathcal{N}_1^{p,q}))$ as a function of $p = 2, \ldots, 20$ and amount of overintegration q

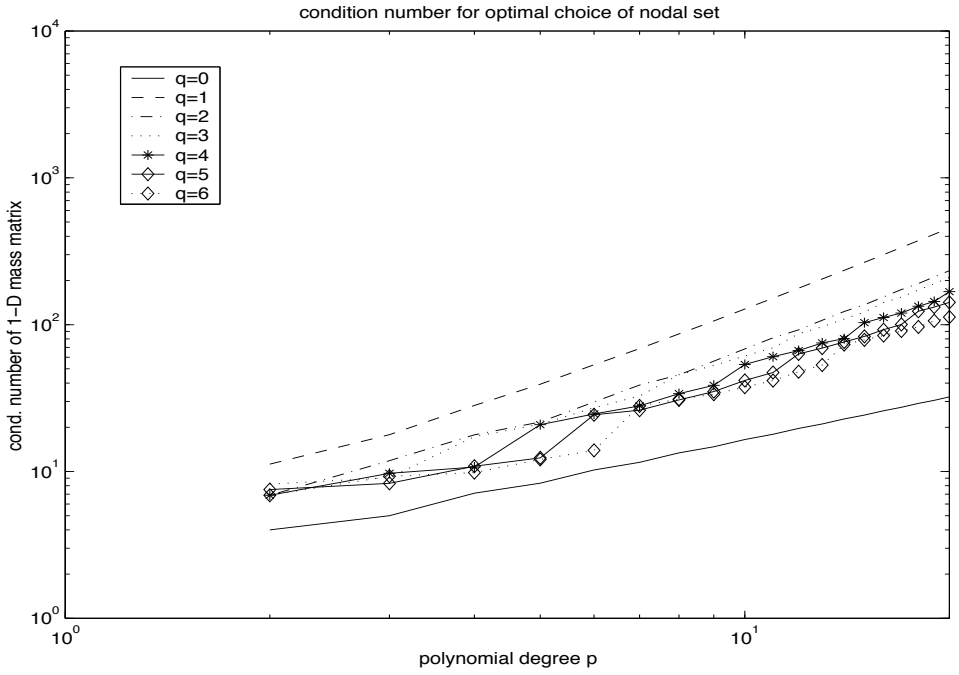

Figure 3: $\mathrm{cond}\,(M^2(\mathcal{N}_2^{p,q}))$ as a function of $p = 2, \ldots, 20$ and amount of overintegration q

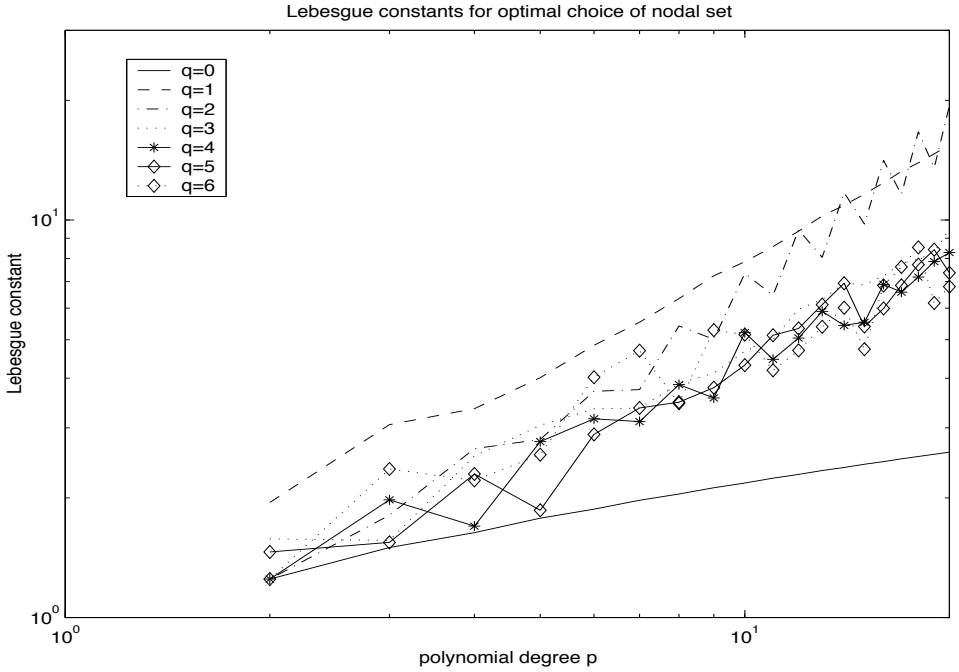

Figure 4: Lebesgue number of set $\mathcal{N}_1^{p,q}$ as a function of $p = 2, \ldots, 20$ and amount of overintegration q

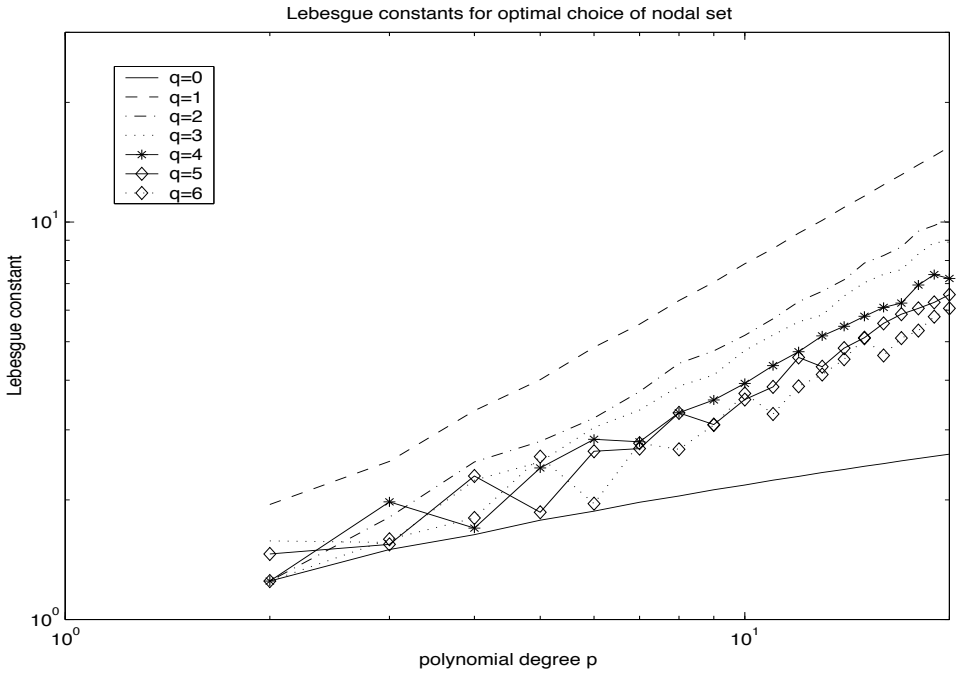

Figure 5: Lebesgue number of set $\mathcal{N}_2^{p,q}$ as a function of $p = 2, \ldots, 20$ and amount of overintegration q

p	2	3	4	5	6	7	8	9	10
q=0	-	-	-	-	-	-	-	-	-
q=2	1,3	1,4	2,4	2,5	2,6	2,7	2,8	2,9	1,11
q=4	1,2,4,5	1,3,4,6	1,3,5,7	1,3,6,8	2,4,6,8	2,4,7,9	2,5,7,10	2,5,8,11	2,6,8,12
q=6	1,2,3,5,6,7	1,2,4,5,7,8	1,3,4,6,7,9	1,2,4,7,9,10	2,3,5,7,9,10	2,3,5,8,10,11	2,4,6,8,10,12	2,3,6,9,12,13	2,5,7,9,11,14

Table 3: index list of $\mathcal{GL}^{p+q} \setminus \mathcal{N}_{1,sym}^{p,q}$.

p	2	3	4	5	6	7	8	9	10
q=0	-	-	-	-	-	-	-	-	-
q=2	1,3	1,4	1,5	2,5	2,6	2,7	3,7	3,8	3,9
q=4	1,2,4,5	1,3,4,6	1,3,5,7	1,3,6,8	1,4,6,9	1,4,7,10	1,4,8,11	2,5,8,11	1,5,9,13
q=6	1,2,3,5,6,7	1,2,4,5,7,8	1,2,4,6,8,9	1,3,5,6,8,10	1,3,5,7,9,11	1,3,5,8,10,12	1,3,6,8,11,13	1,3,6,9,12,14	1,4,7,9,12,15

Table 4: index list of $\mathcal{GL}^{p+q} \setminus \mathcal{N}_{2,sym}^{p,q}$.

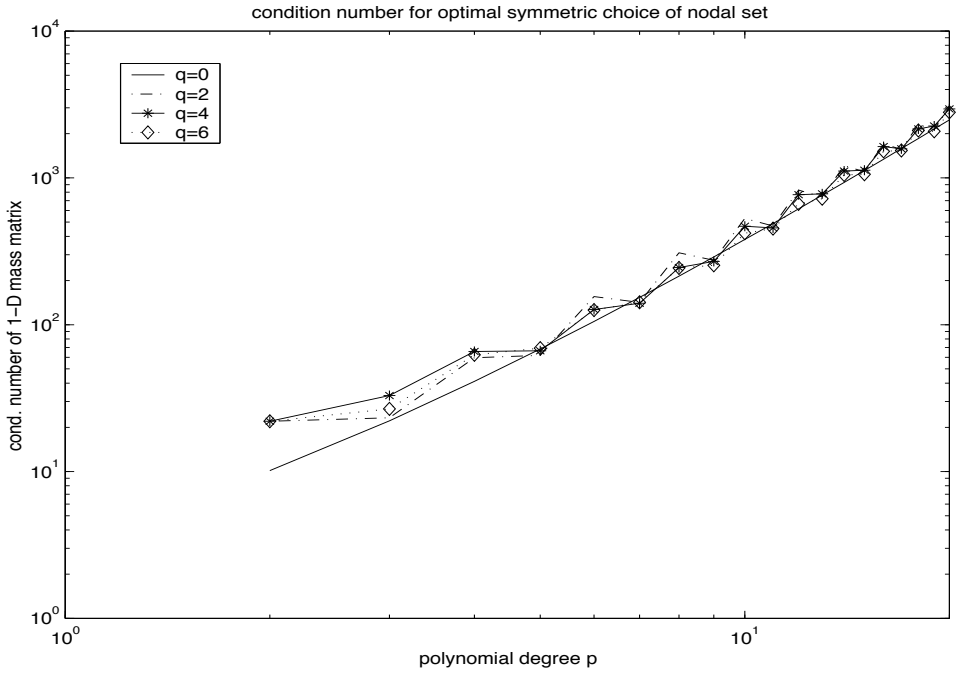

Figure 6: cond $\left(M^1(\mathcal{N}_{1,sym}^{p,q})\right)$ as a function of $p = 2, \ldots, 20$ and amount of overintegration q

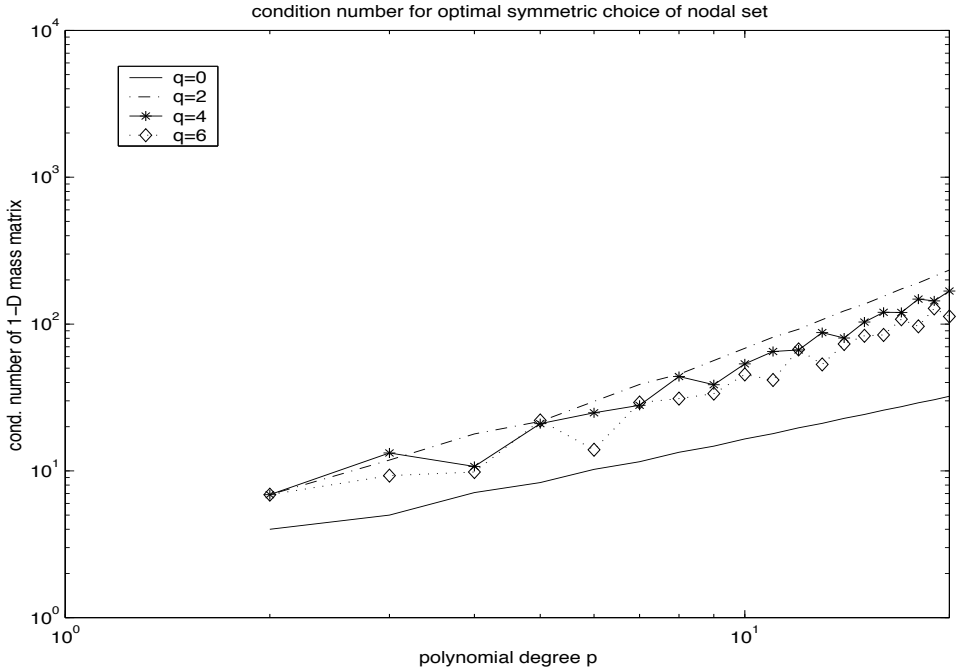

Figure 7: cond $\left(M^2(\mathcal{N}_{2,sym}^{p,q})\right)$ as a function of $p = 2, \ldots, 20$ and amount of overintegration q

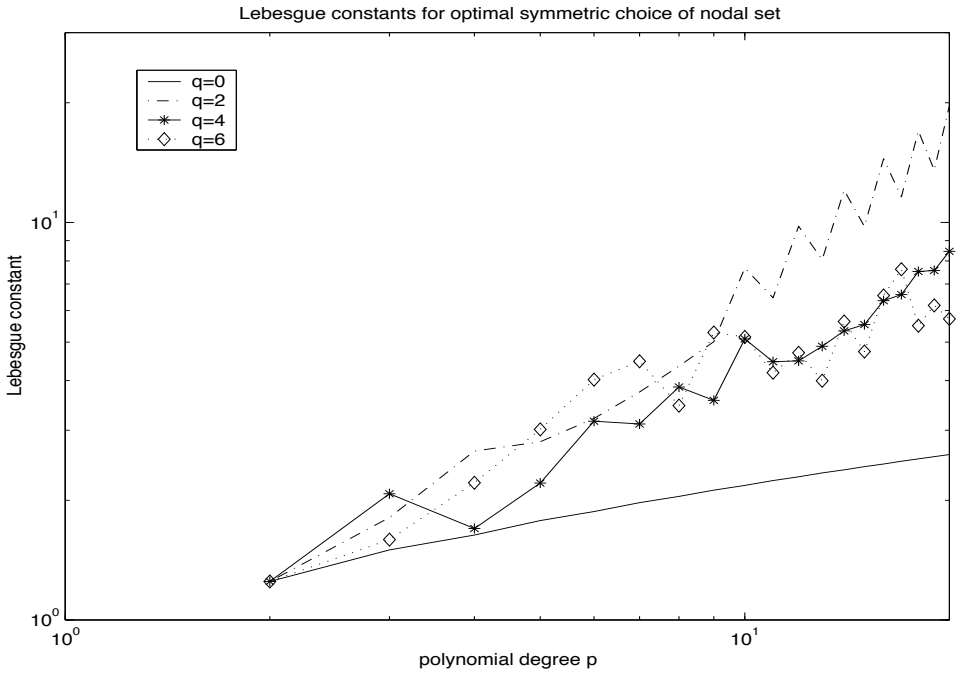

Figure 8: Lebesgue number of set $\mathcal{N}_{1,sym}^{p,q}$ as a function of $p = 2, \ldots, 20$ and amount of overintegration q

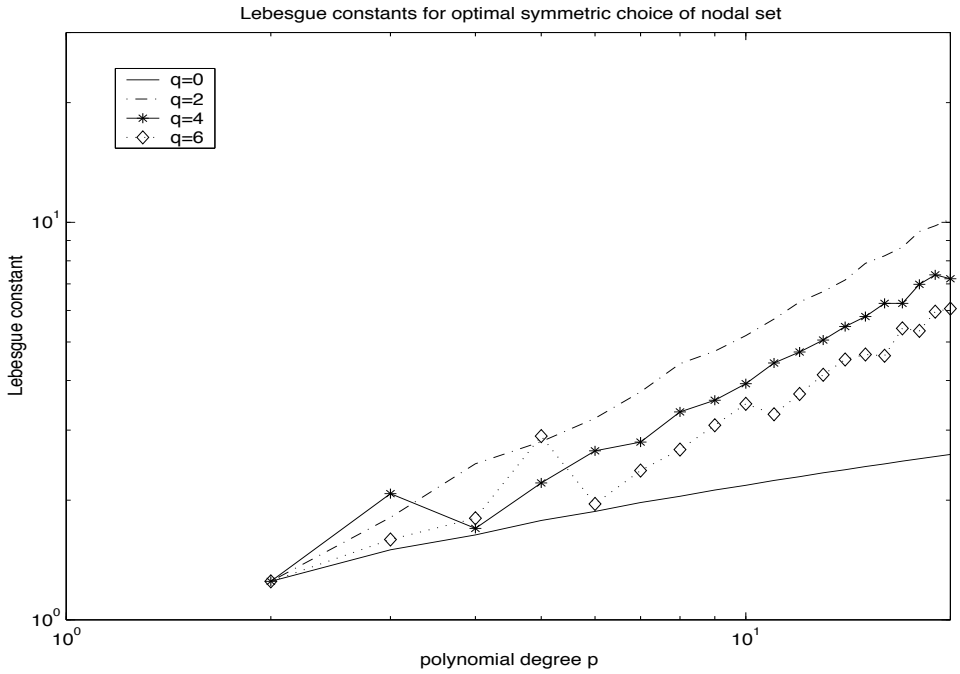

Figure 9: Lebesgue number of set $\mathcal{N}_{2,sym}^{p,q}$ as a function of $p = 2, \ldots, 20$ and amount of overintegration q

2.3.6 Spectral Galerkin Algorithm

We are now in position to combine all the above considerations together to formulate Algorithm 2.12. The algorithm assumes that the space S_{p_K} is of the form (2.9), (2.10). The internal shape functions $\mathcal{I}$ are additionally adapted to the quadrature rule chosen:

Algorithm 2.12 (spectral Galerkin Element Algorithm)
initialize $A_K = 0$
choose external shape functions $\mathcal{E}^k$, $k = 0, \ldots, 4$ for S_{p_K}
determine poly. deg. $p_1 := I^{\mathcal{I}} + 1$, $p_2 := J^{\mathcal{I}} + 1$ of internal shape functions
choose q^h, $q^v \geq 0$ and define quadrature rule $GL_{p_1+q^h, p_2+q^v}$
determine sets $\mathcal{N}^{p_1,q^h}$, $\mathcal{N}^{p_2,q^v}$ according to one of Tables 1, 2, 3, 4
set $\mathcal{I} = \{l_i^{(p_1)}(x, \mathcal{N}^{p_1,q^h}) \cdot l_j^{(p_2)}(y, \mathcal{N}^{p_2,q^v}) \,|\, 1 \leq i \leq p_1 - 1, 1 \leq j \leq p_2 - 1\}$
for r,s=1:2
  for $B_1 \in \{\mathcal{E}^0, \mathcal{E}^1, \mathcal{E}^2, \mathcal{E}^3, \mathcal{E}^4, \mathcal{I}\}$
    for $B_2 \in \{\mathcal{E}^0, \mathcal{E}^1, \mathcal{E}^2, \mathcal{E}^3, \mathcal{E}^4, \mathcal{I}\}$
    add spec_sum_fact$\left(\frac{\partial}{\partial \xi_r} B_1, \frac{\partial}{\partial \xi_s} B_2, \hat{A}_{rs}\right)$ to the block B_1-B_2 of A_K
    end
  end
end

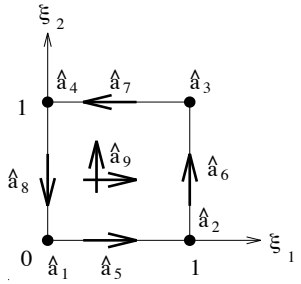

Figure 10: Quadrilateral master element $\hat{K}$ of order $p = (p^1, p^2, p^3, p^4, p^5)$.

Remark 2.13 In Algorithm 2.12 the same quadrature rule is used in the computation of all B_1-B_2 interactions. Of course, the quadrature rule could be chosen individually for each combination, cf. Remark 2.7.

In Algorithm 2.12, we did not specify the external shape functions. Specific choices are the topic of the following subsection.

2.4 *hp* quadrilateral elements

In this section, we present explicitly the shape functions that we use in Algorithm 2.12. In particular, we will illustrate the concept of anisotropic polynomial degree distribution.

We start by introducing the 1-D *hierarchical* shape functions of degree p of [24] as

$$h_1(x) = 1 - x, \quad h_2(x) = x, \quad h_i(x) = \sqrt{\frac{1}{4(2i-1)}}\left(L_i(x) - L_{i-2}(x)\right), \quad i = 3, \dots \tag{2.21}$$

where we recall that the polynomials L_i are the Legendre polynomials associated with the interval $(0, 1)$ normalized to satisfy $L_i(1) = 1$.

For each quadrilateral element $K \in \mathcal{T}$ we introduce the element polynomial degree vector p_K in order to describe the polynomial approximation, i.e

$$p_K = \left(p_K^1, \cdots, p_K^5\right), \tag{2.22}$$

where $p_K^i,\ 1 \le i \le 4$, represents the polynomial degree on the i-th edge of the element, i.e. p_K^i represents the approximation order of the mid-side node $\hat{a}_{i+4}$ in Figure 10. $p_K^5 = (p_K^h, p_K^v)$ is the polynomial degree in the interior of the element which we even allow to be anisotropic, where p_K^h represents the horizontal and p_K^v the vertical approximation order[2]. We only require that

$$p_K^h \ge \max\left\{p_K^1, p_K^3\right\}, p_K^v \ge \max\left\{p_K^2, p_K^4\right\}, \quad \forall\ K \in \mathcal{T}. \tag{2.23}$$

The nodes $\hat{a}_1, \cdots, \hat{a}_9$ represent the dof corresponding to the quadrilateral master element shown in Figure 10 and are divided into three categories:

[2]This feature is essential in the context of thin solids.

1. vertex nodes $\hat{a}_1, \hat{a}_2, \hat{a}_3, \hat{a}_4$: the shape functions associated with the four vertices form the set $\mathcal{E}^0$; they are always taken as the set of the four bilinear functions:

$$\mathcal{E}^0 = \{h_i(x) \cdot h_j(y) \,|\, (i,j) \in \{1,2\} \times \{1,2\}\} \tag{2.24}$$

2. middle node $\hat{a}_9$: the shape functions associated with this node form the set $\mathcal{I}$; they are always taken to be of the form

$$\mathcal{I} = \{l_i^{(p^h)}(x, \mathcal{N}^{p^h,q^h}) \cdot l_j^{(p^v)}(y, \mathcal{N}^{p^v,q^v}) \,|\, 1 \le i \le p^h - 1, \quad 1 \le j \le p^v - 1\} \tag{2.25}$$

here, the sets $\mathcal{N}^{p^h,q^h}$, $\mathcal{N}^{p^v,q^v}$ can be any of those from Tables 1, 2, 3, 4.

3. mid-side nodes $\hat{a}_5, \hat{a}_6, \hat{a}_7, \hat{a}_8$: the shape functions associated with these four nodes form the side shape functions $\mathcal{E}^k$, $k = 1, \ldots, 4$; several possible choices will be given in the following two subsections.

2.4.1 Side shape functions of type (2.17)

The set of *hierarchical* side shape functions of type (2.17) is given by

$$\begin{aligned}
\mathcal{E}^1 &= \{h_i(x) \cdot h_1(y) \,|\, 1 \le i \le p_1 - 1\} && \text{(2.26a)}\\
\mathcal{E}^2 &= \{h_2(x) \cdot h_i(y) \,|\, 1 \le i \le p_2 - 1\} && \text{(2.26b)}\\
\mathcal{E}^3 &= \{h_i(x) \cdot h_2(y) \,|\, 1 \le i \le p_3 - 1\} && \text{(2.26c)}\\
\mathcal{E}^4 &= \{h_1(x) \cdot h_i(y) \,|\, 1 \le i \le p_4 - 1\} && \text{(2.26d)}
\end{aligned}$$

The side shape functions may also be taken as nodal-based functions. Representative for this approach is the following set:

$$\begin{aligned}
\mathcal{E}^1 &= \{l_i^{(p_1)}(x, \mathcal{GL}^{p_1}) \cdot h_1(y) \,|\, 1 \le i \le p_1 - 1\} && \text{(2.27a)}\\
\mathcal{E}^2 &= \{h_2(x) \cdot l_i^{(p_2)}(y, \mathcal{GL}^{p_2}) \,|\, 1 \le i \le p_2 - 1\} && \text{(2.27b)}\\
\mathcal{E}^3 &= \{l_i^{(p_3)}(x, \mathcal{GL}^{p_3}) \cdot h_2(y) \,|\, 1 \le i \le p_3 - 1\} && \text{(2.27c)}\\
\mathcal{E}^4 &= \{h_1(x) \cdot l_i^{(p_4)}(y, \mathcal{GL}^{p_4}) \,|\, 1 \le i \le p_4 - 1\}. && \text{(2.27d)}
\end{aligned}$$

Remark 2.14 The nodal sets $\mathcal{GL}^{p_1}, \ldots,$ may be replaced by other sets, e.g., the sets $\mathcal{N}^{p,q}$. This is, for example, of interest if $p_1 = p^h$ or $p_3 = p^h$ or $p_2 = p^v$ or $p_4 = p^v$ as then (at least some of) the side shape functions are adapted to the quadrature as well, which is then automatically exploited by Algorithm 2.10.

2.4.2 side shape functions of type (2.18)

The side shape functions can of course also be taken of the form (2.18). Specifically, the hierarchical version is given by

$$\begin{aligned}
\mathcal{E}^1 &= \{h_i(x) \cdot l_0^{(p^v)}(y, \mathcal{N}^{p^v,q^v}) \,|\, 1 \le i \le p_1 - 1\} && \text{(2.28a)}\\
\mathcal{E}^2 &= \{l_{p^h}^{(p^h)}(x, \mathcal{N}^{(p^h),q^h}) \cdot h_i(y) \,|\, 1 \le i \le p_2 - 1\} && \text{(2.28b)}\\
\mathcal{E}^3 &= \{h_i(x) \cdot l_{p^v}^{(p^v)}(y, \mathcal{N}^{p^v,q^v}) \,|\, 1 \le i \le p_3 - 1\} && \text{(2.28c)}\\
\mathcal{E}^4 &= \{l_0^{(p^h)}(x, \mathcal{N}^{p^h,q^h}) \cdot h_i(y) \,|\, 1 \le i \le p_4 - 1\} && \text{(2.28d)}\\
& && \text{(2.28e)}
\end{aligned}$$

where the sets $\mathcal{N}^{p^h,q^h}$, $\mathcal{N}^{p^v,q^v}$ are now chosen from Table 2 or 4. Here, p^h, p^v, q^h, q^v are determined by the internal shape functions and the quadrature rule. Note that these side shape functions are (at least partially) adapted to the quadrature rule which is automatically exploited by Algorithm 2.10.

Analogously, the version using Gauss-Lobatto side shape functions is given by

$$\mathcal{E}^1 = \{l_i^{(p_1)}(x, \mathcal{GL}^{p_1}) l_0^{(p^v)}(y, \mathcal{N}^{p^v,q^v}) \,|\, 1 \le i \le p_1 - 1\} \tag{2.29a}$$

$$\mathcal{E}^2 = \{l_{p^h}^{(p^h)}(x, \mathcal{N}^{p^h,q^h}) l_i^{(p_2)}(y, \mathcal{GL}^{p_2}) \,|\, 1 \le i \le p_2 - 1\} \tag{2.29b}$$

$$\mathcal{E}^3 = \{l_i^{(p_3)}(x, \mathcal{GL}^{p_3}) l_{p^v}^{(p^v)}(y, \mathcal{N}^{p^v,q^v}) \,|\, 1 \le i \le p_3 - 1\} \tag{2.29c}$$

$$\mathcal{E}^4 = \{l_0^{(p^h)}(x, \mathcal{N}^{p^h,q^h}) l_i^{(p_4)}(y, \mathcal{GL}^{p_4}) \,|\, 1 \le i \le p_4 - 1\}. \tag{2.29d}$$

Again, as in the case (2.27), the sets $\mathcal{GL}^{p_1}, \ldots, \mathcal{GL}^{p_4}$ could be replaced with sets of the form $\mathcal{N}^{p,q}$.

Remark 2.15 Note that the vertex shape functions are always taken as the bilinear shape functions. This is motivated by implementational issues. Clearly, other choices are possible.

3 Extensions to 3D

3.1 Work estimates in 3D

The ideas presented for the 2D model problem can be extended to 3D. Table 5 collects the work estimates for the 2D and the 3D variants of the standard quadrature Algorithm 2.2, the sum factorization Algorithm 2.5, and the spectral-Galerkin Algorithm 2.10. In this subsection, we will briefly point at some of the differences between 2D and 3D and illustrate how the work estimates of Table 5 were obtained.

A common decomposition of the spaces S_{p_K} is into vertex shape functions, edge shape functions, face shape functions, and internal shape functions with the following two properties: first, all shape functions have tensor product structure and may be represented in the following fashion (note that a 3D hexahedron has 8 vertices, 12 edges, and 6 faces for a total of 26 geometric entities)

$$\begin{aligned} \mathcal{E}^k &= \{\chi_{1,i}^m(\xi_1)\chi_{2,j}^m(\xi_2)\chi_{3,k}^m(\xi_3)\,|, 1 \le i \le I_m,\ 1 \le j \le J_m,\ 1 \le k \le K_m\},\ m = 1, \ldots, 26, \\ \mathcal{I} &= \{\chi_{1,i}^{\mathcal{I}}(\xi_1)\chi_{2,j}^{\mathcal{I}}(\xi_2)\chi_{3,k}^{\mathcal{I}}(\xi_3)\,|, 1 \le i \le I_{\mathcal{I}}, \quad 1 \le j \le J_{\mathcal{I}}, \quad 1 \le k \le K_{\mathcal{I}}\}. \end{aligned}$$

Second, the numbers I_m, J_m, K_m satisfy

$$\forall m \in \{1, \ldots, 26\} \quad \text{at least one of the numbers } I_m,\ J_m,\ K_m \text{ is equal to 1.} \tag{3.30}$$

In order to fix ideas, we will again consider for our work estimates the case $Q_p(\hat{K})$. The estimates, however, are valid for the general case as well. In that case, the values I_m, J_m,

K_m, and $I_{\mathcal{I}}$, $J_{\mathcal{I}}$, $K_{\mathcal{I}}$ would typically satisfy the following conditions:

for the 8 vertices, $m \in \{1,\dots,8\}$:	$I_m = J_m = K_m = 1$
for the 12 edges, $m \in \{9,\dots,20\}$:	exactly one of the three numbers I_m, J_m, K_m equals $p-1$ and the other two equal 1
for the 6 faces, $m \in \{21,\dots,26\}$:	exactly two of the three numbers I_m, J_m, K_m equal $p-1$ and the remaining one equals 1
for $\mathcal{I}$:	$I_{\mathcal{I}} = J_{\mathcal{I}} = K_{\mathcal{I}} = p-1$.

Since the shape functions have tensor product structure, the sum factorization Algorithm 2.5 can be employed to evaluate the stiffness matrix with an operation count of $O(p^4(p+q)^3)$ (the use of $GL_{p+q,p+q,p+q}$ is assumed). Before proceeding to the analysis of the 3D analogon of Algorithm 2.12 we point out that the calculation of all the $\mathcal{E}^m$-$\mathcal{E}^n$ blocks of the stiffness matrix is done with $O(p^4(p+q) + p^2(p+q)^2 + p(p+q)^3)$ work by Algorithm 2.5. This operation count is due to the fact that our version of sum factorization re-orders the triple sums of the quadrature rule so as to minimize the work. Assumption (3.30) ensures that there is always one ordering of the triple sums which yields this operation count (which is $O(p^5)$ for fixed q). If no reordering is performed, it is easy to see that sum factorization techniques perform the calculation of the $\mathcal{E}^m$-$\mathcal{E}^n$ blocks in $O(p^6)$ complexity. This re-ordering is therefore essential for the efficiency of our *hp*-spectral Galerkin algorithm in 3D.

We now turn to the work estimates for the $\mathcal{I}$-$\mathcal{E}$ and $\mathcal{I}$-$\mathcal{I}$ blocks with internal shape functions that are adapted to the quadrature rule. In the 3-D version of Algorithm 2.10, 6 possible reorderings of the triple quadrature sum have to be checked. However, in order to obtain work estimates for the 3-D version of Algorithm 2.10, it suffices to consider the floating point operation count for one particular, judiciously chosen ordering. We illustrate this with an example, the calculation of $\texttt{spec\_sum\_fact}(\frac{\partial}{\partial\xi_1}\mathcal{I}, \frac{\partial}{\partial\xi_2}\mathcal{E}^{21}, a)$. Here, $\mathcal{E}^{21}$ is a set of face shape functions satisfying $I_{21} = J_{21} = p-1$, $K_{21} = 1$. We evaluate the triple quadrature sum as

$$
\begin{aligned}
A_{ijk,i'j'k'} &= \sum_{q_2} \chi^{\mathcal{I}}_{2,j}(x_{2,q_2}) \left(\chi^{21}_{2,j'}\right)'(x_{2,q_2}) H_1(q_2,k,k',i,i'),\\
\text{where } H_1(q_2,k,k',i,i') &= \sum_{q_3} \chi^{\mathcal{I}}_{3,k}(x_{3,q_3}) \chi^{21}_{3,k'}(x_{3,q_3}) H_2(q_2,q_3,i,i'),\\
\text{and } H_2(q_2,q_3,i,i') &= \sum_{q_1} \left(\chi^{\mathcal{I}}_{1,i}\right)'(x_{1,q_1}) \chi^{21}_{1,i'}(x_{1,q_1}) a(x_{1,q_1},x_{2,q_2},x_{3,q_3}) w_{1,q_1} w_{2,q_2} w_{3,q_3}.
\end{aligned}
$$

Generating the auxiliary array H_2 can be achieved with work $W(H_2) = O((p+q)^2p^2(p+q))$. Next, as the internal shape functions are adapted to the quadrature rule, the sum in the definition of H_1 has only $q+1$ non-vanishing entries for each k, and we arrive at $W(H_1) = O((p+q)p^3(1+q))$ for the generation of H_1. Computing the final sum, we can again exploit that the internal shape functions are adapted to the quadrature rule to obtain $W = O(p^3p^2(1+q))$. The total cost is therefore $W(A) = O(p^5(1+q)+q^2p^3+q^3p^2)$. We note that the essential feature of the above summation order is that the two outer sums have only $1+q$ non-zero terms and that only the innermost sum has $p+q+1$ non-zero terms; for our second-order model problem, the triple quadrature sums can always be arranged in this way for the $\mathcal{I}$-$\mathcal{E}$ blocks.

Let us consider now the calculation of the $\mathcal{I}$-$\mathcal{I}$ block. We consider `spec_sum_fact`$(\frac{\partial}{\partial\xi_1}\mathcal{I}, \frac{\partial}{\partial\xi_1}\mathcal{I}, a)$. This is evaluated as

$$\begin{aligned} A_{ijk,i'j'k'} &= \sum_{q_2} \chi^{\mathcal{I}}_{2,j}(x_{2,q_2})\chi^{\mathcal{I}}_{2,j'}(x_{2,q_2})H_1(q_2,k,k',i,i'), \\ H_1(q_2,k,k',i,i') &= \sum_{q_3} \chi^{\mathcal{I}}_{3,k}(x_{3,q_3})\chi^{\mathcal{I}}_{3,k'}(x_{3,q_3})H_2(q_2,q_3,i,i') \\ H_2(q_2,q_3,i,i') &= \sum_{q_1} \left(\chi^{\mathcal{I}}_{1,i}\right)'(x_{1,q_1})\left(\chi^{\mathcal{I}}_{1,i'}\right)'(x_{1,q_1})a(x_{1,q_1},x_{2,q_2},x_{3,q_3})w_{1,q_1}w_{2,q_2}w_{3,q_3}. \end{aligned}$$

We check that generating the auxiliary array H_2 costs $W(H_2) = O((p+q)^3p^2)$. Next, for the evaluation of the array H_1 it is convenient to write the set of quadrature points $\mathcal{GL}^{p+q}$ as $\mathcal{GL}^{p+q} = \mathcal{N}^{p,q} \cup \left(\mathcal{GL}^{p+q} \setminus \mathcal{N}^{p,q}\right)$ and express H_1 as

$$\begin{aligned} H_1 &= \sum_{x\in\mathcal{N}^{p,q}} \chi^{\mathcal{I}}_{3,k}(x)\chi^{\mathcal{I}}_{3,k'}(x)H_2 + \sum_{x\in\mathcal{GL}^{p+q}\setminus\mathcal{N}^{p,q}} \chi^{\mathcal{I}}_{3,k}(x)\chi^{\mathcal{I}}_{3,k'}(x)H_2 \\ &= \delta_{k,k'}H_2(q_2,k,i,i') + \sum_{x\in\mathcal{GL}^{p+q}\setminus\mathcal{N}^{p,q}} \chi^{\mathcal{I}}_{3,k}(x)\chi^{\mathcal{I}}_{3,k'}(x)H_2. \end{aligned}$$

The first term can be evaluated with work $W = O((p+q)p^3)$. The sum has only q terms and it costs $W = O(q(p+q)p^4)$ flops. The cost for the evaluation of H_1 is therefore $W(H_1) = O(p^4 + qp^5 + q^2p^4)$. Completely analogously, we conclude that the outermost sum is performed with work $W = O(qp^6 + p^5)$. The total cost for the $\mathcal{I} - \mathcal{I}$ block is therefore $W(A) = O(qp^6 + p^5 + q^2p^4 + q^3p^2)$. Hence, the $(p+1)^6$ entries of the stiffness matrix can be computed in optimal complexity $O(p^6)$.

Remark 3.1 It is interesting to note that the case of "minimal" quadrature, $q = 0$, is special: The leading order term qp^6 vanishes, and the complexity reduces again to $O(p^5)$ [3]. This reduction of the complexity for $q = 0$ is due to the fact that adapting the internal shape functions to the minimal quadrature rule introduces a significant amount of *sparsity* in the $\mathcal{I}$-$\mathcal{I}$ block of the stiffness matrix as only $O(p^5)$ entries of this block of size $O(p^6)$ are non-zero.

We collect the work estimates for the standard Algorithm 2.2, the sum factorization Algorithm 2.5, and our Algorithm 2.12 for the quadrature rule $GL_{p+q,p+q,p+q}$ in conjunction with the space $Q_p(\hat{K})$ in Table 5. For Algorithm 2.12, we included in Table 5 only the work for the computation of the non-zero entries of the stiffness matrix.

3.2 Work estimates in 3D: numerical example

In the present section, we illustrate the quadrature speed-up of Algorithm 2.12 over both the standard quadrature Algorithm 2.2 and the sum factorization Algorithm 2.5. All

[3]The work estimates in Table 5 ignore the initialization phase $A_K = 0$, which is also an $O(p^6)$ algorithm. To obtain a truely $O(p^5)$ algorithm for the case $q = 0$, a special storage format for A_K has to be used, that stores the non-zero entries only. From a practical point of view, one can assume that initializing $A_K = 0$ is done very efficiently by the operating system so that the $O(p^6)$ contribution is negligible in the practical regime of polynomial degrees p.

algorithm	2D	3D
standard	$O(p^4(p+q)^2)$	$O(p^6(p+q)^3)$
sum fact.	$O(p^4(p+q)+p^2(p+q)^2)$	$O(p^6(p+q)+p^4(p+q)^2+p^2(p+q)^3)$
spect.-Gal.	$O(p^4(1+q)+p^2q^2)$	$O(qp^6+p^5+q^2p^4+q^3p^2)$

Table 5: Asymptotic flop estimates for various quadrature schemes with overintegration of order $q \geq 0$

three algorithms were used to generate the stiffness matrix for a 3D scalar Poisson problem with zero-order term (hence, the mass matrix has to be generated also) with variable coefficients for the space $Q_p(\hat{K})$, $p = 1, \ldots, 9$. The actual shape functions are taken as the 3D analogs of (2.24) for the vertex shape functions, of (2.25) for the internal shape functions, and of (2.26) for the edges and faces. The stiffness matrix calculation was done with the 3D-version of the *hp*-code HP90, [8], a general *hp*-code written in Fortran 90; actual calculations were performed on a SUN UltraSparc running at 336Mhz using the manufacturer provided F90 compiler with optimization flag ("-fast") set. In Fig. 11 we compare three algorithms: the standard Algorithm 2.2 (with Gauss-Legendre quadrature with $(p+1)^3$ points), the sum factorization Algorithm 2.5 (again with Gauss-Legendre quadrature with $(p+1)^3$ points) and Algorithm 2.12 with Gauss-Lobatto quadrature rule $GL_{p,p,p}$ (i.e., $q = 0$). For comparison reasons, we additionally show the time needed to condense out the internal degrees of freedom as for environments where static condensation is performed as part of the element stiffness matrix generation routine, this determines a lower bound for the speed-up. Fig. 11 in fact contains the timings for two different condensation schemes: the faster of the two (labelled "blas3 condensation") uses the manufacturer-optimized LAPACK routine `dposv` and the BLAS3 routine `dgemm`. For the element generation, the Spectral-Galerkin algorithm outperforms both the sum-factorization and the standard algorithm: For $p = 9$, the stiffness matrix is generated in 228 secs. by the standard algorithm, in 46 secs. by the sum factorization and in only 19.6 secs. by the Spectral-Galerkin algorithm. The regular static condensation takes 28 secs. whereas the optimized one takes 2.7 secs. Thus, for our scalar model problem and the range $p = 1, \ldots, 9$, our Spectral-Galerkin algorithm does provide a means to significantly speed up the element CPU-time: compared with (even our improved version of) the sum factorization algorithm, the Spectral-Galerkin algorithm generates the condensed element matrix by a factor 2 faster ($46 + 2.7$ secs. vs. $19.6 + 2.7$ secs.).

Comparing the slopes of the curves in Fig. 11, we also see that the Spectral-Galerkin algorithm is of lower complexity than the sum factorization algorithm showing that our asymptotic complexity analysis above is valid for small p as well.

A detailed break-down of the timings for the generation of the different blocks of the stiffness matrix for the sum factorization technique and the Spectral-Galerkin Algorithm is done in Figs. 12, 13. Adapting the internal shape functions to the quadrature rule has the biggest impact on the timings for the $\mathcal{I}$-$\mathcal{I}$ block (the "bubble-bubble" block): Whereas for the sum factorization (cf. Fig. 12) it is the most expensive part, it is the cheapest for the Spectral-Galerkin algorithm in the range $p = 2$—9 (cf. Fig. 13). Some speed-up is also visible for the generation of the bubble-face block. Note, however, that the timings here are given for our sum factorization Algorithm 2.5, which is an improved version of the classical algorithm in that re-ordering of the quadrature sums is performed. Otherwise, the

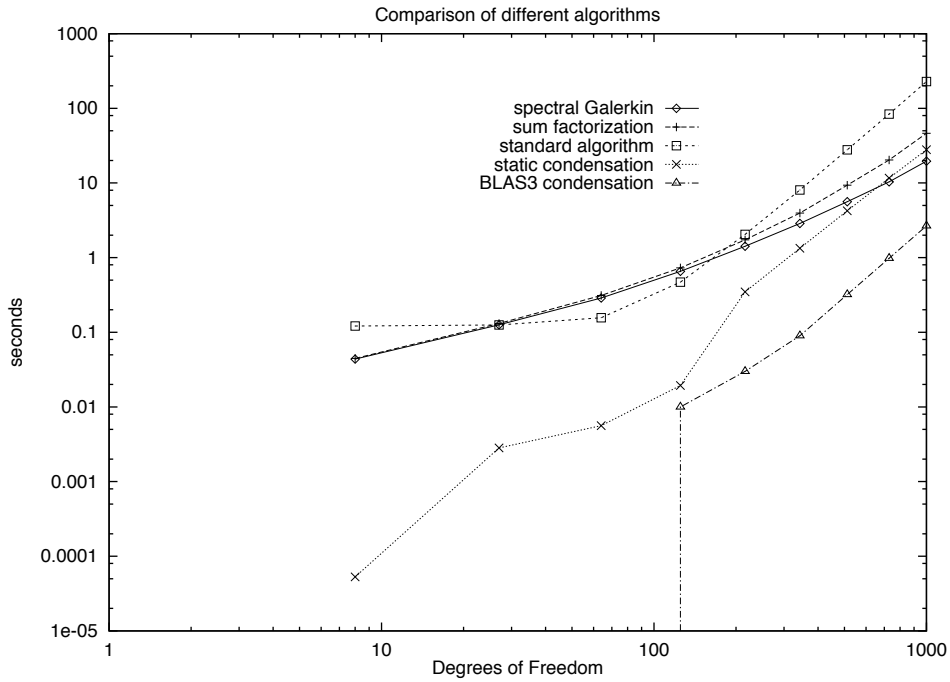

Figure 11: CPU time per hexahedral element for standard, sum factorization, and spectral-Galerkin algorithm for scalar 3D model problem; also included: standard condensation and condensation based on optimized LAPACK and BLAS routines.

discrepancy between the classical sum factorization algorithm and the Spectral-Galerkin algorithm for the bubble-face block would have been bigger. As the Spectral-Galerkin algorithm reduces to sum factorization techniques for the $\mathcal{E}^n$-$\mathcal{E}^m$ blocks, the timings for the blocks not involving the internal shape functions are identical.

Fig. 14 finally shows the timings for various auxiliary and book-keeping procedures: the evaluation of the 1-D shape functions in the quadrature points, dealing with constraint nodes, and the computation of the right hand side are seen to be negligible compared to the quadrature.

In our algorithms, we did not focus on the cost of evaluating the geometry and the coefficient matrix at the quadrature points. In our calculations, the actual element map was a trilinear function and the coefficient matrix A was very simple, thus leading to an operation count of $O((p+q)^3)$ for sampling these data in the quadrature points. The timings for this part of the element computation are given in Fig. 14 and labelled "eval. Jacobian". We point out that this part of the computation does depend strongly on the actual element maps (or approximations thereof) and the coefficients of the differential equations.

4 Conclusions

We have presented and discussed quadrature techniques for *hp*-FEM in two and three dimensional elliptic problems. The main idea consisted in the design of shapefunctions that are adapted to the quadrature rule and in the classical sum-factorization used. A special case is the spectral element method which is well-established in computational

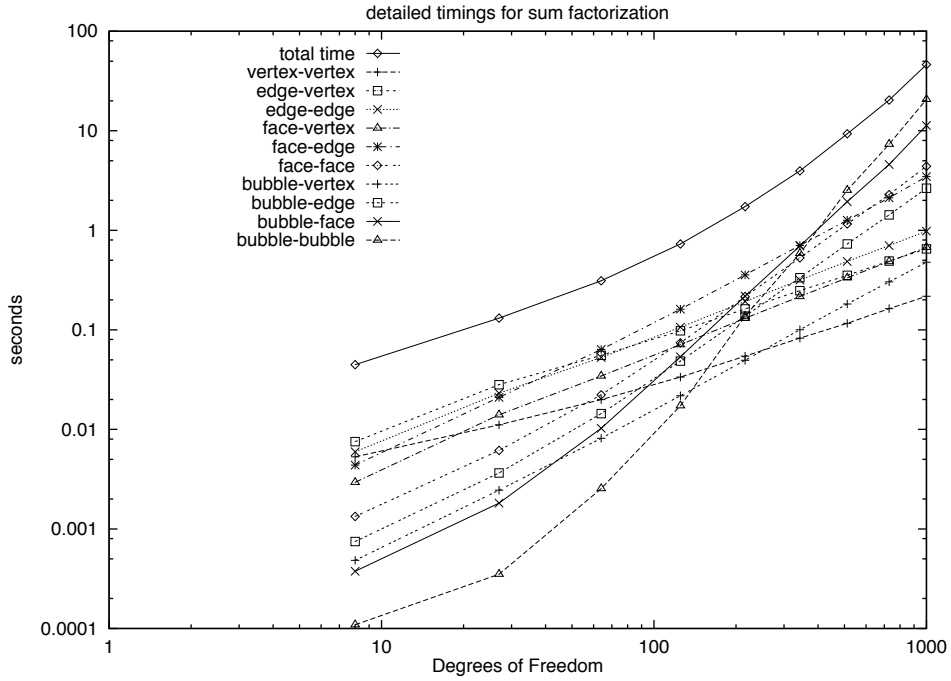

Figure 12: Break-down of timing for sum factorization algorithm for scalar 3D model problem.

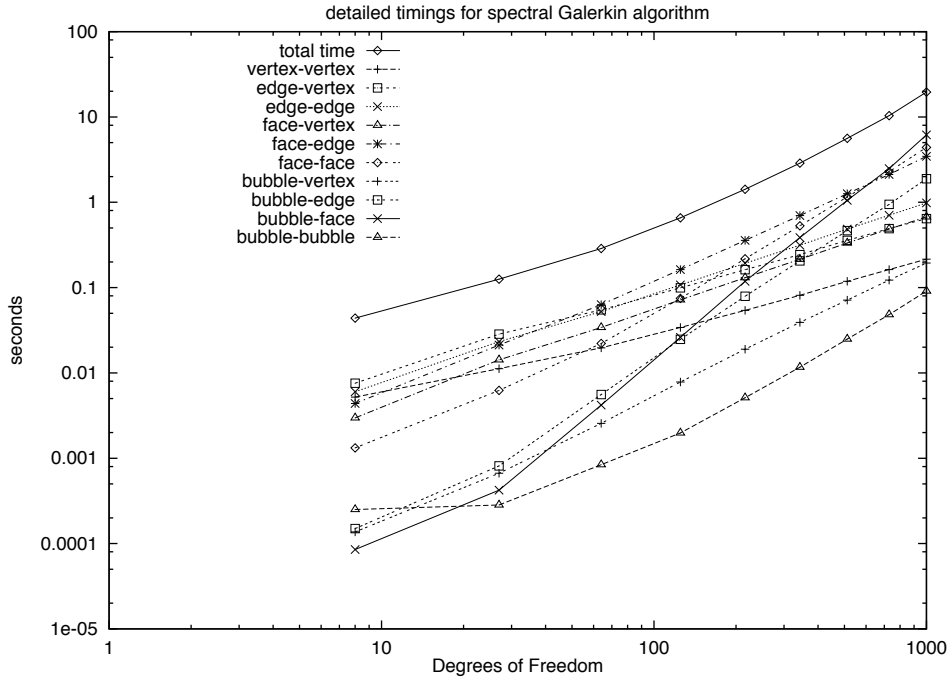

Figure 13: Break-down of timing for Spectral-Galerkin algorithm for scalar 3D model problem.

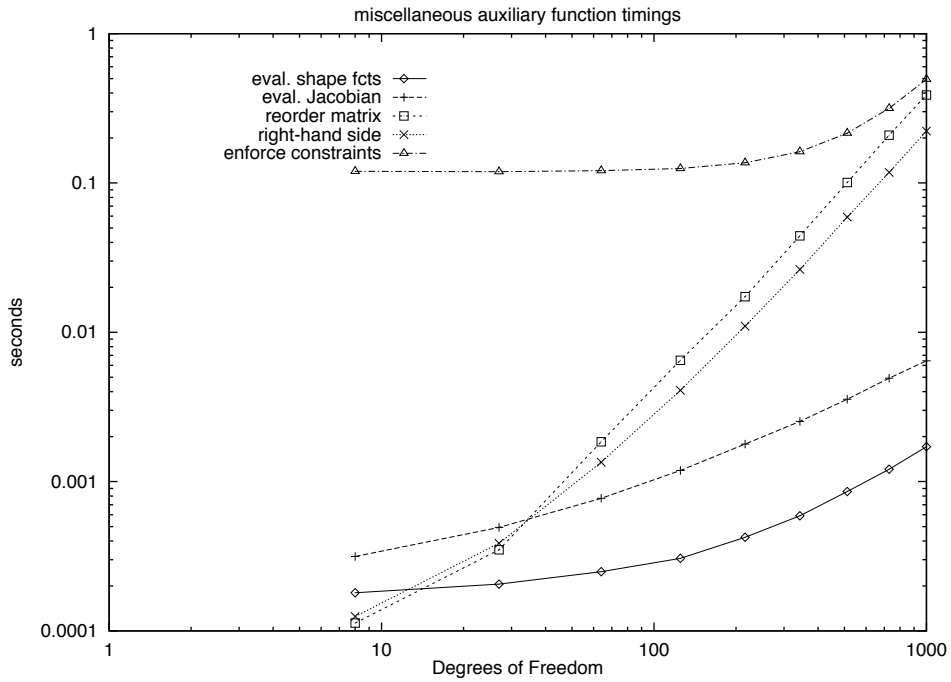

Figure 14: Timings for auxiliary functions sum factorization/spectral Galerkin algorithm for scalar 3D model problem.

fluid dynamics. The spectral element method is well-known, however, to underintegrate the element stiffness matrices which is known to cause stability problems if elliptic systems are thus discretized. We generalized therefore the ideas in the spectral element method by decoupling the quadrature order from the elemental degree, as is customary in Galerkin *hp*-FEM. Lagrange shapefunctions based on a suitable subset of the Gauss-Lobatto nodes were proposed. New choices of nodal sets for these functions were given based on the criterion of small condition number for the resulting stiffness matrices. The condition numbers thus obtained were found to be marginally worse than those of the spectral element method, while the CPU-times for the element stiffness matrix generation were comparable to those of the spectral element method. The new algorithm was found to outperform the standard Galerkin scheme with sum-factorization. For polynomial degrees between 1 and 10, static condensation was found to be considerably cheaper than the quadrature for the stiffness matrix generation. The ideas in the present paper have natural generalizations to triangles and tetrahedra which will be presented elsewhere.

Acknowledgement: We thank Philipp Frauenfelder for implementing the 3D quadrature rules in the code HP90, [8].

References

[1] R. Actis, B. Szabó, C. Schwab, Hierarchic models for laminated plates and shells, *Comput. Meths. Appl. Mech. Eng.* vol. 172 (1999), p.79–107

[2] I. Babuška and M. Suri. The *p* and *hp* Versions of the Finite Element Methods, Basic Principles and Properties. *SIAM review* **36** (1994) 578-632.

[3] C. Bernardi and Y. Maday. *Approximations spectrales de problèmes aux limites elliptiques*. Springer, 1992.

[4] F. Ben Belgacem, Y. Maday. A spectral element methodology tuned to parallel implementations. *Comput. Meths. Appl. Mech. Eng.* **116** (1994), p. 59–67.

[5] C. Canuto, M. Y. Hussaini, A. Quarteroni and T. A. Zhang. *Spectral Methods in Fluid Dynamics*, Springer Verlag, 1986.

[6] L. Demkowicz, J.T. Oden, W. Rachowicz and O. Hardy. Toward a Universal *hp* Adaptive Finite Element Strategy. Part 1: Constrained Approximation and Data Structure. *Computer Methods in Applied Mechanics and Engineering*, 77 (1989), 79-112.

[7] L. Demkowicz, W. Rachowicz, K. Banas and J. Kucwaj. 2-D *hp* Adaptive Package (2D*hp*AP). *Technical Report*, Section of Applied Mathematics, Technical University of Cracow, Poland, 1992.

[8] L. Demkowicz, K. Gerdes, C. Schwab, A. Bajer and T. Walsh. HP90: A general and flexible Fortran 90 *hp*-FE code. *Computing and Visualization in Science* 1 (1998) 145-163.

[9] K. Gerdes, J.M. Melenk, and C. Schwab. Fully Discrete *hp*-FEM: Error Analysis. (in preparation)

[10] D. Gottlieb and S.A. Orszag, Numerical Analysis of Spectral Methods: Theory and Applications, CBMS 26, SIAM, 1977.

[11] W. Hackbusch. *Multigrid Methods and Applications.* Springer Verlag, 1985.

[12] G.E. Karniadakis and S.J. Sherwin, Spectral/hp Element Methods for CFD, Oxford University Press, 1999

[13] K.Z. Korczak and A.T. Patera, An Isoparametric Spectral Element Method for Solution of the Navier-Stokes Equations in Complex Geometry. *J. Comput. Phys.* **62** (1986), p. 361–382

[14] Y. Maday and A.T. Patera. Spectral Element Method for Navier-Stokes Equations. In: *State of the Art Surveys in Computational Mechanics* (A.K. Noor and J.T. Oden, eds.), (1989), p. 71–143

[15] P. Le Tallec and A. Patra. Non-overlapping domain decomposition methods for adaptive *hp* approximations of the Stokes problem with discontinuous pressure fields. *Comput. Methods Appl. Mech. Engrg.* 145 (1997) 361-379.

[16] J. T. Oden, A. Patra and Y. Feng. Parallel Domain Decomposition Solver for Adaptive *hp* Finite Element Methods. *SIAM J. Numer. Anal.* vol. 34 (1997) 2090-2118.

[17] S.A. Orszag, spectral methods for problems in complex geometries, J. Comput. Phys., vol. 37 (1980), 70–92.

[18] A.T. Patera, Spectral Element Method for Fluid Dynamics: Laminar Flow in a Channel Epansion, *J. Comput. Phys.* **54** (1984), p. 568–488

[19] C. Schwab. The p and hp FEM. Oxford University Press, 1998.

[20] STRIPE, Flygtekniska Försöksanstalten, Box 11021, S-16111 Bromma, Sweden.

[21] Stresscheck, ESRD Inc., 10845 Olive Boulevard, Suite 170, St. Louis, MO 63141-7760, `http://www.esrd.com`

[22] PHLEX, Computational Mechanics Company, Inc. 7800 Shoal Creek Blvd, Suite 290E, Austin, TX 78757-1024, `http://www.comco.com`

[23] Burkhard Sündermann. Lebesgue Constants in Lagrangian Interpolation at the Fekete points. Ergebnisberichte der Lehrstühle Mathematik III und VIII (Angewandte Mathematik) 44, Universität Dortmund, 1980.

[24] B. Szabó and I. Babuška. *Finite Element Analysis*. Wiley 1991.

Research Reports

No.	Authors	Title
99-15	J.M. Melenk, K. Gerdes, C. Schwab	Fully Discrete *hp*-Finite Elements: Fast Quadrature
99-14	E. Süli, P. Houston, C. Schwab	*hp*-Finite Element Methods for Hyperbolic Problems
99-13	E. Süli, C. Schwab, P. Houston	*hp*-DGFEM for Partial Differential Equations with Nonnegative Characteristic Form
99-12	K. Nipp	Numerical integration of differential algebraic systems and invariant manifolds
99-11	C. Lage, C. Schwab	Advanced boundary element algorithms
99-10	D. Schötzau, C. Schwab	Exponential Convergence in a Galerkin Least Squares *hp*-FEM for Stokes Flow
99-09	A.M. Matache, C. Schwab	Homogenization via *p*-FEM for Problems with Microstructure
99-08	D. Braess, C. Schwab	Approximation on Simplices with respect to Weighted Sobolev Norms
99-07	M. Feistauer, C. Schwab	Coupled Problems for Viscous Incompressible Flow in Exterior Domains
99-06	J. Maurer, M. Fey	A Scale-Residual Model for Large-Eddy Simulation
99-05	M.J. Grote	Am Rande des Unendlichen: Numerische Verfahren für unbegrenzte Gebiete
99-04	D. Schötzau, C. Schwab	Time Discretization of Parabolic Problems by the hp-Version of the Discontinuous Galerkin Finite Element Method
99-03	S.A. Zimmermann	The Method of Transport for the Euler Equations Written as a Kinetic Scheme
99-02	M.J. Grote, A.J. Majda	Crude Closure for Flow with Topography Through Large Scale Statistical Theory
99-01	A.M. Matache, I. Babuška, C. Schwab	Generalized *p*-FEM in Homogenization
98-10	J.M. Melenk, C. Schwab	The *hp* Streamline Diffusion Finite Element Method for Convection Dominated Problems in one Space Dimension
98-09	M.J. Grote	Nonreflecting Boundary Conditions For Electromagnetic Scattering
98-08	M.J. Grote, J.B. Keller	Exact Nonreflecting Boundary Condition For Elastic Waves
98-07	C. Lage	Concept Oriented Design of Numerical Software
98-06	N.P. Hancke, J.M. Melenk, C. Schwab	A Spectral Galerkin Method for Hydrodynamic Stability Problems