ELSEVIER

Comput. Methods Appl. Mech. Engrg. 190 (2001) 4339–4364

Computer methods in applied mechanics and engineering

www.elsevier.com/locate/cma

Fully discrete *hp*-finite elements: fast quadrature

J.M. Melenk [a], K. Gerdes [b], C. Schwab [c,*]

[a] *Max-Planck-Institut für Mathematik in den Naturwissenschaften, Inselstr. 22–26, D-04103, Leipzig, Germany*
[b] *Department of Mathematics, Chalmers University, SE-41296, Göteborg, Sweden*
[c] *Seminar für Angewandte Mathematik, ETH Zürich, CH-8092, Zürich, Switzerland*

Received 4 October 1999

Abstract

A fully discrete *hp*-finite element method (FEM) is presented. It combines the features of the standard *hp*-FEM (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. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: hp-finite element method; Spectral element method; Numerical integration

1. Introduction

The finite element method (FEM) is the most widely used discretization technique in solid and fluid mechanics. The classical approach still dominant in industrial applications today 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, $1 \leqslant p \leqslant 3$). Convergence is achieved through mesh refinement, i.e., by letting $h \to 0$.

In the early 1980s, the so-called *p*-version FEM emerged in which the polynomial degree p is increased on a fixed mesh in order to produce convergent sequences of FE-solutions (see e.g. [3,25] and the references therein). At the same time, spectral methods and spectral element methods (cf. [4,6,13,17,19,20] together with the references therein) were developed. These methods are closely related to the *p*-version FEM and can often be interpreted as *p*-version FEMs with certain types of quadrature.

Many problems in fluid and solid mechanics have piecewise analytic solutions. For the approximation of such functions, the *h*-version, *p*-version FEM, and 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 FEMs 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 an increase of the polynomial degree in regions where the solution is smooth (analytic) then leads to *exponential* rates of convergence (see [3,7,8,23]). For spectral methods, the observed need for mesh refinement has led to various non-conforming approaches, notably, ''mortar elements'' (see e.g. [5] and the references therein), which also exhibit *hp*-like features.

* Corresponding author.
E-mail addresses: melenk@mis.mpg.de (J.M. Melenk), schwab@sam.math.ethz.ch (C. Schwab).

PII: S0045-7825(00)00322-4

From an approximation theoretical point of view, the flexibility of the *hp*-FEM to combine *h*-refinement and *p*-enrichment makes it clearly superior to more traditional *h*-, *p*-version FEMs, and spectral methods. Computationally, however, *hp*-FEMs are often perceived as expensive in their stiffness matrix generation and lacking fast solution algorithms. In contrast, the stiffness matrix generation in *h*-FEMs is comparatively simple and efficient multilevel solvers (e.g. [14]) are available. Spectral methods have the reputation of being fast owing to techniques such as sum factorization and weighted collocation (instead of quadrature). Additionally, a variety of preconditioners for spectral methods are available (see e.g. [10,22]). While the present paper focuses on the fast stiffness matrix generation in *hp*-FEM, we mention that other computational aspects of *hp*-FEM such as the design of efficient preconditioners (see e.g. [1,12,21]) and parallelization (e.g. [16,18]) are currently active research areas.

This paper addresses the efficient stiffness matrix generation in *hp*-FEM. We develop the *hp*-spectral Galerkin FEM, which unifies and generalizes "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 local mesh refinement via hanging nodes. The element stiffness matrices are obtained completely in parallel. Likewise, local static condensation can be performed completely in parallel. Geometry representation is decoupled from the finite-element space, thus accommodating exact representations of the geometry via blending methods and the use of *hp*-isoparametric elements that approximate the geometry. We draw on techniques from spectral methods, in particular sum factorization and adapting the shape functions to the quadrature rule. Generalizing the spectral quadrature to polynomial degrees p and quadrature rules of order $p+q, q \geqslant 0$ arbitrary, we show that the work per element is $\mathrm{O}(p^4(1+q))$ and $\mathrm{O}(qp^6+p^5)$ for problems in $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively (see Table 5 ahead for the details). [1] These work estimates are to be compared with $\mathrm{O}(p^{3d})$ 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 $5 \leqslant p \leqslant 10$.

We finally mention that similar algorithms have been proposed in, e.g. [15] 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 2.13) is presented. For clarity of exposition, the algorithm is presented for a 2-D scalar model problem. The extension to higher dimensions is discussed in Section 3.1. Section 3.2 demonstrates for a 3-D hexahedral element that already for moderate polynomial degrees p, Algorithm 2.13 leads to significant computational savings. Some closing remarks conclude the paper.

2. *hp*-spectral quadrature algorithm

2.1. Model problem

We will illustrate the *hp*-spectral Galerkin algorithm (Algorithm 2.13) for the following scalar model problem in two dimensions

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

where the matrix A is symmetric, positive definite, and piecewise smooth. The weak form is:

Find $u \in H_0^1(\Omega)$ such that

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

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

[1] Here and in what follows the implied constant in the $\mathrm{O}(\cdot)$ notation is independent of both p and q.

$$a(u,v)=\int_\Omega \nabla u\cdot(A(x)\nabla v)\,\mathrm{d}x\,\mathrm{d}y,\quad l(v)=\int_\Omega fv\,\mathrm{d}x\,\mathrm{d}y.$$

We restrict our attention to the model problem (2.1) for simplicity of notation. The *hp*-spectral Galerkin algorithm can be adapted in a straightforward manner to 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 $\mathscr{T}=\{K\}$. 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 sums of element integrals, and the element maps F_K provide a means to express all integrals over elements K as integrals over the master element $\hat{K}$. The FEM can then equivalently be written as the following problem:

Find $u\in S$ s.t.

$$\sum_{K\in\mathscr{T}}a_K(u,v)=\sum_{K\in\mathscr{T}}l_K(v)\quad\forall v\in S,\quad a_K(u,v)=\int_{\hat{K}}\hat{\nabla}\hat{u}\cdot(\hat{A}\hat{\nabla}\hat{v})\,\mathrm{d}\xi_1\,\mathrm{d}\xi_2=\int_{\hat{K}}\sum_{i,j=1}^{2}\frac{\partial\hat{u}}{\partial\xi_i}\hat{A}_{ij}\frac{\partial\hat{v}}{\partial\xi_j}\,\mathrm{d}\xi_1\,\mathrm{d}\xi_2$$

$$=:\sum_{i,j=1}^{2}a_K^{ij}(u,v),\quad l_K(u,v)=\int_{\hat{K}}\hat{f}\hat{v}\,\mathrm{d}\xi_1\,\mathrm{d}\xi_2,$$

$$\hat{A}=\left(F_K'\right)^{-T}(A\circ F_K)\left(F_K'\right)^{-1}J_K, \tag{2.4}$$

$$\hat{f}=f\circ F_K J_K, \tag{2.5}$$

$$\hat{u}=u\circ F_K,\quad \hat{v}=v\circ F_K$$

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

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

are called 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 assembled 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. This brief description of the generic *hp*-Galerkin FEM can be condensed into the following algorithm:

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 coefficients in (2.1) are constant and 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 [2], are employed instead. For an error analysis of such approximate mappings we refer to [11].

The most elementary quadrature algorithm for the generation of the element stiffness matrix is

Algorithm 2.2 (*Standard Galerkin element quadrature algorithm*).

```
initialize A_K = 0
for all quadrature points
  for all basis functions φ of S_pK
    for all basis functions ψ of S_pK
      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 on the element level, namely, the quadrature using Algorithm 2.2, the static condensation of internal degrees of freedom, the enforcement of constraints, and the assembly. The CPU time is plotted versus the polynomial degree p ranging from $p = 1$ to $p = 8$. The finite-element space $S_{p_K}(\hat{K})$ is taken as the tensor product space Q_p, the space of polynomials of degree p in each variable

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

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. 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 2.13 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 2.13 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 without any condensation is possible. Then, as in spectral and spectral element methods, Algorithm 2.13 can be reformulated to realize a matrix vector multiplication rather than to generate explicitly the (local) stiffness matrices.

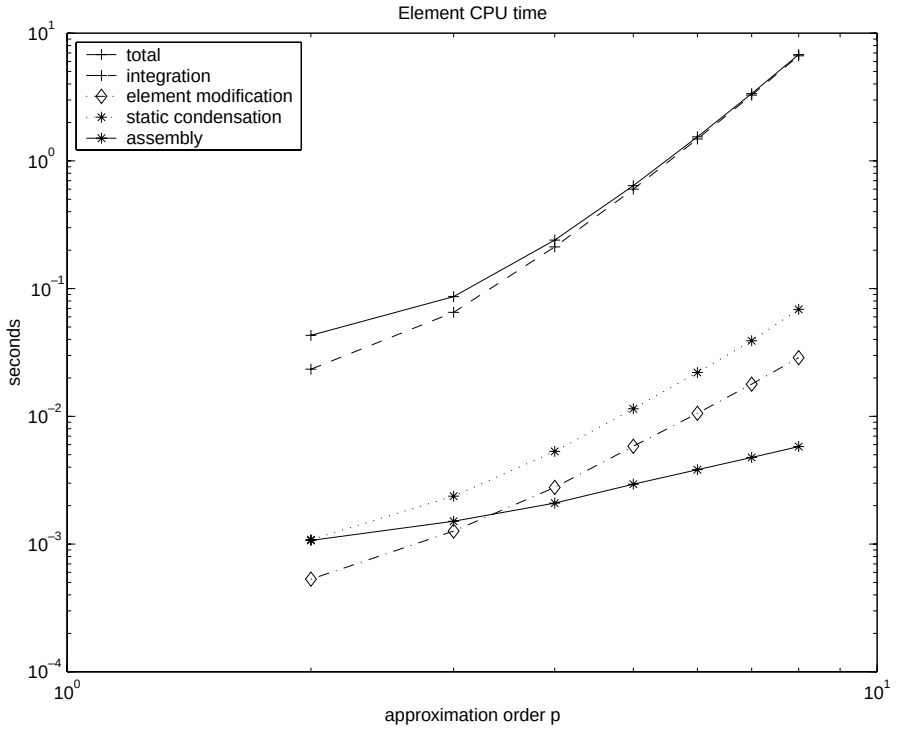

Fig. 1. CPU time per quadrilateral element for standard element algorithms for scalar 2-D model problem.

2.3. Element stiffness matrix generation

The goal of this section is to derive Algorithm 2.13, 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.

In order to derive Algorithm 2.13, we start by recalling Gauss–Lobatto quadrature in Section 2.3.1. Next, we describe a modified sum factorization technique (Algorithm 2.5). Algorithm 2.10, which is at the heart of Algorithm 2.13, 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, we will always give an operation count for the case $S_{p_K}(\hat{K}) = Q_p(\hat{K})$.

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 $\mathscr{G}L^q = \{x_0, \ldots, x_q\}$ are the zeros of the polynomial $x \mapsto x(1-x)L'_q(x)$, where L_q stands for qth Legendre polynomial associated with the unit interval $(0,1)$. It is well known [4] 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

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

is exact for polynomials u of degree $2q-1$ [4]. Gauss–Lobatto quadrature rules in two dimensions are then obtained by tensor product constructions. For example, the quadrature rules $GL_{q_1,q_2}(u)$ based on the points $\mathscr{G}L^{q_1} \times \mathscr{G}L^{q_2}$ 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 the 1-D quadrature points are sorted in ascending order; in particular, we have $x_{1,0} = 0, x_{1,q_1} = 1, x_{2,0} = 0$, $x_{1,q_2} = 1$.

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 $\hat{K}$ with the quadrature rule (2.7). For example, for $\varphi, \psi \in S_{p_K}(\hat{K})$ 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 $S_{p_K}(\hat{K}) = \operatorname{span}\{\mathscr{E}^0 \cup \mathscr{E}^1 \cup \mathscr{E}^2 \cup \mathscr{E}^3 \cup \mathscr{E}^4 \cup \mathscr{I}\}$, where the sets $\mathscr{E}^k, \mathscr{I}$ have the following tensor product structure:

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

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

This structure is motivated by the usual decomposition into *vertex shape functions*, *side shape functions*, and *internal shape functions*. In that case, the numbers $I_k, J_k, I_{\mathscr{I}}, J_{\mathscr{I}}$ 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)

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

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

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

The element stiffness matrix A_K has a 6×6 block structure, corresponding to the interaction of the six different types of shape functions introduced in (2.9) and (2.10). Using sum factorization, Algorithm 2.4 exploits the tensor product structure of the basis functions:

Algorithm 2.4.

initialize $A_K = 0$
for r, s = 1:2
  for $B_1 \in \{\mathscr{E}^0, \mathscr{E}^1, \mathscr{E}^2, \mathscr{E}^3, \mathscr{E}^4, \mathscr{I}\}$
    for $B_2 \in \{\mathscr{E}^0, \mathscr{E}^1, \mathscr{E}^2, \mathscr{E}^3, \mathscr{E}^4, \mathscr{I}\}$
      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 $\partial/\partial\xi_r$, $\partial/\partial\xi_s$ 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
  %$A_{(i,j),(i',j')} = \sum_{k=0}^{q_1}\sum_{l=0}^{q_2} \varphi_i^1(x_{1,k})\varphi_j^2(x_{2,l})\psi_{i'}^1(x_{1,k})\psi_{j'}^2(x_{2,l})w_{1,k}w_{2,l}a(x_{1,k},x_{2,l})$

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

Remark 2.6. Algorithm 2.5 is essentially the classical sum factorization algorithm (e.g. [19]), which is used in several commercial 3-D *hp*-FEM codes. The new feature of Algorithm 2.5 is the comparison of S_1 and S_2. S_1 and S_2 estimate the operation count for the evaluation of the quadrature 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. The work estimates S_1 and S_2 in Algorithm 2.5 could of course be replaced with more refined measures for the computational work, e.g., measures that factor in specific computer hardware such memory access times; one could also envisage look-up tables for S_1 and S_2 obtained from actual test runs.

Remark 2.7. An additional reduction of the computational work could be achieved by choosing an individual quadrature rule for each pair B_1–B_2. To illustrate this point, let us note that a typical choice for the vertex set $\mathscr{E}^0$ is the set of the four bilinear functions and that the set of internal shape functions $\mathscr{I}$ 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 sum_fact($\mathscr{E}^0,\mathscr{I}$,a) whereas $GL_{p,p}$ is needed for the calculation in sum_fact($\mathscr{I},\mathscr{I}$,a).

Example 2.8 (*Work for Algorithm* 2.5). For the simplified case $q_1 = q_2 = p+q$, $q \ge 0$, inspection of Algorithm 2.5 shows that its 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 2-D model case $S_{p_K}(\hat{K}) = Q_p(\hat{K})$, we can express this more explicitly in terms of p and q. Assuming that the sets $\mathscr{I}$ and $\mathscr{E}^k$ satisfy (2.11)–(2.14) reveals that, for example, the call sum_fact($(\partial/\partial\xi_1)\mathscr{I}$, $(\partial/\partial\xi_2)\mathscr{I}, \hat{A}_{12}$) takes work $W = \mathrm{O}(p^4(p+q) + p^2(p+q)^2)$. Similarly, the calls sum_fact($(\partial/\partial\xi_1)\mathscr{I}$, $(\partial/\partial\xi_2)\mathscr{E}^1, \hat{A}_{12}$) and sum_fact($(\partial/\partial\xi_1)\mathscr{E}^1, (\partial/\partial\xi_2)\mathscr{E}^2, \hat{A}_{12}$) are of complexity $W = \mathrm{O}(p^3(p+q) + p(p+q)^2)$ and $W = \mathrm{O}(p(p+q)^2 + p^2(p+q))$, 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 Lagrange interpolation polynomials 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 $L(j,j') := \{l \in \{0,\dots,q_2\} \,|\, \varphi_j^2(x_{2,l})\psi_{j'}^2(x_{2,l}) \neq 0\}$
$S_1 := (1+q_1)\sum_{j,j'} |\, L(j,j') \,| + J \cdot J' \cdot \sum_{i,i'} |\, K(i,i') \,|$ % work estimate for $\sum_k^{q_1}\sum_l^{q_2}$
$S_2 := (1+q_2)\sum_{i,i'} |\, K(i,i') \,| + I \cdot I' \cdot \sum_{j,j'} |\, L(j,j') \,|$ % work estimate for $\sum_l^{q_2}\sum_k^{q_1}$
if $S_1 \le S_2$ then { % choose summation order to minimize work estimate
    compute auxiliary array $H(k,j,j') := \sum_{l \in L(j,j')} \varphi_j^2(x_{2,l})\psi_{j'}^2(x_{2,l})w_{1,k}w_{2,l}a(x_{1,k},x_{2,l})$,
    compute entries $A_{(i,j),(i',j')} := \sum_{k \in K(i,i')} \varphi_i^1(x_{1,k})\psi_{i'}^1(x_{1,k})H(k,j,j')$,
  }
else {
    compute auxiliary array $H(l,i,i') := \sum_{k \in K(i,i')} \varphi_j^1(x_{1,k})\psi_{j'}^1(x_{1,k})w_{1,k}w_{2,l}a(x_{1,k},x_{2,l})$,
    compute entries $A_{(i,j),(i',j')} := \sum_{l \in L(j,j')} \varphi_j^2(x_{2,l})\psi_{j'}^2(x_{2,l})H(l,i,i')$,
}
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 reduces 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 leads 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 $\mathscr{E}^k$ and $\mathscr{I}$ are assumed to satisfy (2.9) and (2.10). The internal shape functions $\mathscr{I}$ are additionally assumed to be in the form

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

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

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

We note that there holds

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

Let us compute the cost of calling $\texttt{spec\_sum\_fact}((\partial/\partial\xi_1)\mathscr{I}, (\partial/\partial\xi_2)\mathscr{E}^1, \hat{A}_{12})$ under the assumption that the shape functions comprising $\mathscr{E}^1$ are not adapted to the quadrature rule (recall, however, that the shape functions of $\mathscr{I}$ are adapted to the quadrature rule). The sets $K(i,i')$, $L(j,j')$ are then seen to be

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

Inspection of Algorithm 2.10 reveals that this function call has complexity $W = \mathrm{O}(p^4)$. For the evaluation of $\texttt{spec\_sum\_fact}((\partial/\partial\xi_1)\mathscr{I}, (\partial/\partial\xi_2)\mathscr{I}, \hat{A}_{12})$, we note that

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

Hence, this evaluation is also accomplished with work $W = \mathrm{O}(p^4)$. The reader will convince himself that for any $r,s \in \{1,2\}$, $k \in \{0,\dots,4\}$, the calls $\texttt{spec\_sum\_fact}((\partial/\partial\xi_r)\mathscr{I}, (\partial/\partial\xi_s)\mathscr{I}, \hat{A}_{rs})$ and $\texttt{spec\_sum\_fact}((\partial/\partial\xi_r)\mathscr{I}, (\partial/\partial\xi_s)\mathscr{E}^k, \hat{A}_{rs})$ are performed with work $W = \mathrm{O}(p^4)$.

Remark 2.12. Example 2.11 is in essence the classical spectral method. In its classical form, the shape functions $\mathscr{E}^k$ are also adapted to the quadrature, which, while not improving the order of complexity of the algorithm, reduces the constant in the work estimate $\mathrm{O}(p^4)$.

We note that using the Gauss–Lobatto rule $GL_{p,p}$ leads to *underintegration* because, even for affine elements, the stiffness matrix is not computed exactly. For our scalar problem, we show in [11] that the rule $GL_{p,p}$ in conjunction with the space $Q_p(\hat{K})$ leads to a stable method; furthermore [11] shows that for piecewise analytic input data the exponential rate of convergence of the Galerkin FEM with exact quadrature is preserved. Nevertheless, this "minimal" spectral quadrature scheme is known to perform poorly in the presence of very distorted meshes. To the knowledge of the authors, stability of the spectral element method (i.e., the use of the rule $GL_{p,p}$) applied to vector-valued problems such as the Lamé equations on meshes containing non-affine elements is still an open problem. These considerations make it desirable to allow for *overintegration*, i.e., the ability to use the rule $GL_{p+q,p+q}$ ($q \geqslant 0$) in conjunction with the space $Q_p(\hat{K})$. We will show in the ensuing subsection that this is possible with work $W = \mathrm{O}(p^4(1+q) + p^2q^2)$.

2.3.4. Transition from spectral to Galerkin via overintegration

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

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

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

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

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

$$\left|\left\{x_j \in \mathscr{G}L^{p+q} | l_i^{(p)}(x_j, \mathcal{N}^p) \neq 0\right\}\right| = |(\{i\} \cup \mathscr{G}L^{p+q} \setminus \mathcal{N}^p)| = 1+q, \quad i = 1,\dots,p-1.$$

In this situation, let us calculate the cost of evaluating $\texttt{spec\_sum\_fact}((\partial/\partial\xi_1)\mathscr{I}, (\partial/\partial\xi_2)\mathscr{E}^2, \hat{A}_{12})$. We see that

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
$q=0$	–	–	–	–	–	–	–	–	–
$q=1$	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, 12
$q=5$	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 2
Index lists of $\mathcal{GL}^{p+q} \setminus \mathcal{N}_2^{p,q}$

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

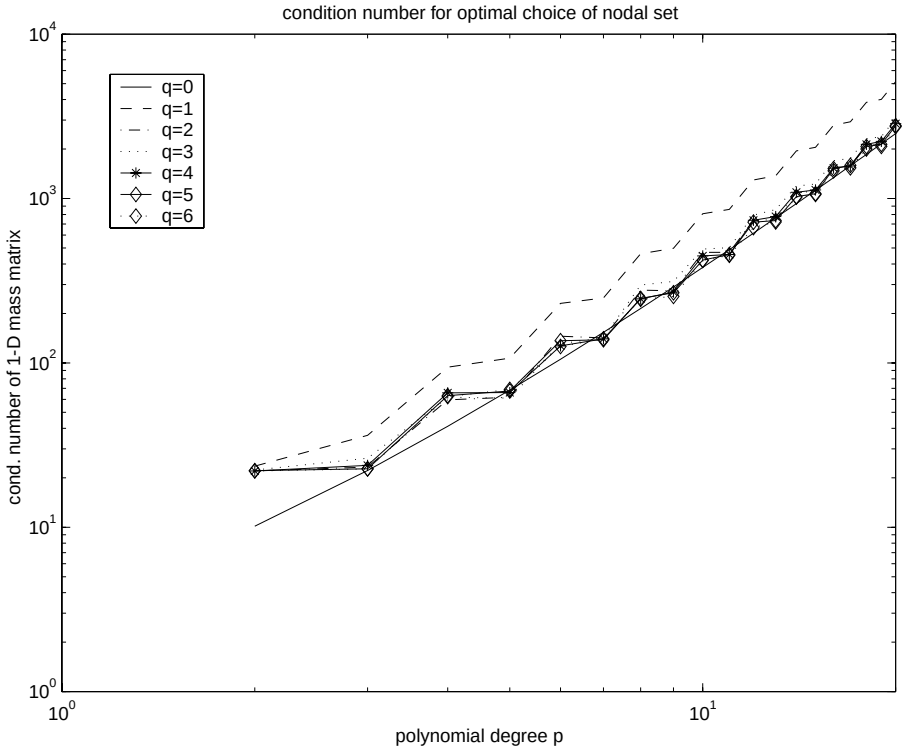

Fig. 2. $\mathrm{cond}(M^1(\mathcal{N}_1^{p,q}))$ as a function of $p = 2, \ldots, 20$ and degree of overintegration q.

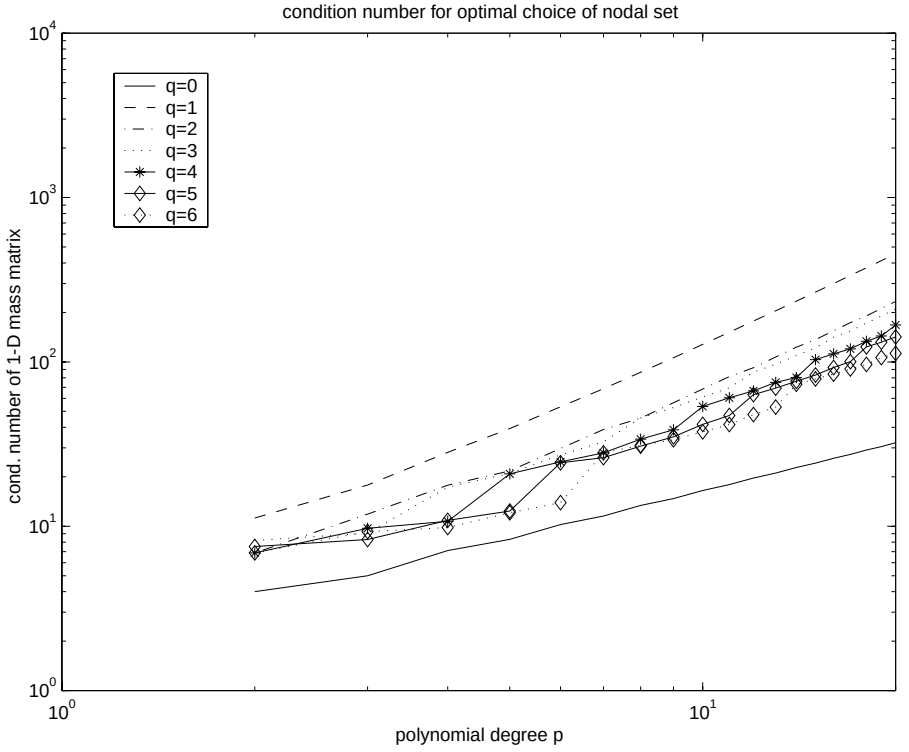

Fig. 3. $\mathrm{cond}(M^2(\mathcal{N}_2^{p,q}))$ as a function of $p = 2, \ldots, 20$ and degree of overintegration q.

$$|K(i,i')| = |\mathscr{G}L^{p+q}| = |\{0,\dots,p+q\}| = p+q+1, \quad |L(j,j')| = |(\{j\} \cup (\mathscr{G}L^{p+q} \setminus \mathscr{N}^p))| = 1+q.$$

Inspection of Algorithm 2.10 reveals a work estimate $W = \mathrm{O}((p+q)p(p^2+q))$. Similarly, for calling `spec_sum_fact`$((\partial/\partial\xi_1),\mathscr{I},(\partial/\partial\xi_2),\mathscr{I},\hat{A}_{12})$ we get

$$|K(i,i')| = \left|\left(\{i'\} \cup \mathscr{G}L^{p+q} \setminus \mathscr{N}^p\right)\right| = q+1, \quad |L(j,j')| = |(\{j\} \cup (\mathscr{G}L^{p+q} \setminus \mathscr{N}^p))| = q+1.$$

Thus, the work can be bounded by $W = \mathrm{O}(p^2(p+q)(1+q)) + \mathrm{O}(p^4(q+1)) = \mathrm{O}(p^4(1+q) + p^2q^2))$. We conclude that in two dimensions, the use of the Gauss–Lobatto rule of order $p+q$ with fixed $q \geqslant 0$ has the same asymptotic order of complexity as the pure spectral method of Example 2.11.

We presented work estimates for the isotropic case of the rule $GL_{p+q,p+q}$ together with the space $Q_p(\hat{K})$. Similar arguments apply for more general (e.g., anisotropic) polynomial degree distribution and anisotropic quadrature rules.

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

In Section 2.3.4, we merely assumed the interpolation nodes $\mathscr{N}^p$ to be a subset of the quadrature points $\mathscr{G}L^{p+q}$. In this section, we propose to use the condition number of a 1-D mass matrix as the main criterion for selecting the interpolation nodes $\mathscr{N}^p$.

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

$$\chi_0^1(x) := 1-x, \quad \chi_p^1(x) := x, \quad \chi_i^1(x) := l_i^{(p)}(x,\mathscr{N}^p), \quad i = 1,\dots,p-1, \tag{2.17}$$

$$\chi_i^2(x) := l_i^{(p)}(x,\mathscr{N}^p), \quad i = 0,\dots,p. \tag{2.18}$$

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

Minimization of the 1*-D mass matrix*. For each of the two sets of shape functions (2.17) and (2.18), we can define the *mass matrix* M^m ($m \in \{1,2\}$) in the usual fashion

$$M_{ij}^m(\mathscr{N}^p) := \int_0^1 \chi_i^m(x)\chi_j^m(x)\,\mathrm{d}x, \quad i,j = 0,\dots,p+1.$$

The two mass matrices $M^m(\mathscr{N}^p), m \in \{1,2\}$, depend of course on the nodal set $\mathscr{N}^p$ through the shape functions χ_i^m, and we included the argument $\mathscr{N}^p$ in the definition of M^m in order to emphasize this dependence. Given $p \geqslant 1, q \geqslant 0$, we seek a subset $\mathscr{N}^p$ of $\mathscr{G}L^{p+q}$ such that the mass matrix M^m ($m \in \{1,2\}$) has minimal condition number. The corresponding *optimal* choice of the nodal set is then denoted by $\mathscr{N}_1^{p,q}$ for the shape functions (2.17) and by $\mathscr{N}_2^{p,q}$ for the shape functions (2.18)

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

Tables 1 and 2 give the optimal sets $\mathscr{N}_1^{p,q}, \mathscr{N}_2^{p,q}$ (up to symmetry); in the interest of brevity, Tables 1and 2 list the indices of the sets $\mathscr{G}L_{p+q} \setminus \mathscr{N}_m^{p,q}$ ($m \in \{1,2\}$) rather than the indices of the sets $\mathscr{N}_m^{p,q}$. Figs. 2 and 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; hence, 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 $\mathscr{N}^p$ is its Lebesgue number $\Lambda(\mathscr{N}^p)$, which is the stability constant of the corresponding nodal interpolation operator

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

Figs. 4 and 5 show the Lebesgue numbers for the 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 and 5), is essentially the best possible choice with $\Lambda(\mathcal{GL}^p) = \mathrm{O}(\ln p)$, [24]. 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 $\Lambda(\mathcal{GL}^p)$.

Symmetry of point distribution. We note that in both Tables 1 and 2, the optimal points are not distributed symmetrically with respect to the midpoint $1/2$. For implementational purposes during the assembly procedure, it might be 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 $1/2$

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

The optimal sets are listed in Tables 3 and 4 (again, Tables 3 and 4 actually list the indices of $\mathcal{GL}_{p+q} \setminus \mathcal{N}_{m,sym}^{p,q}$). Figs. 6 and 7 show the corresponding condition numbers. Comparing these figures with Figs. 2 and 3 we note that imposing a symmetry condition on the nodal sets $\mathcal{N}^p$ has practically 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 and 9 with Figs. 4 and 5, respectively.

2.3.6. Spectral Galerkin algorithm

We are now in a position to combine all above considerations to formulate Algorithm 2.13. The algorithm assumes that basis functions of the space S_{p_K} can be chosen in forms (2.9) and (2.10). The main feature of Algorithm 2.13 is that the internal shape functions are adapted to the chosen quadrature rule.

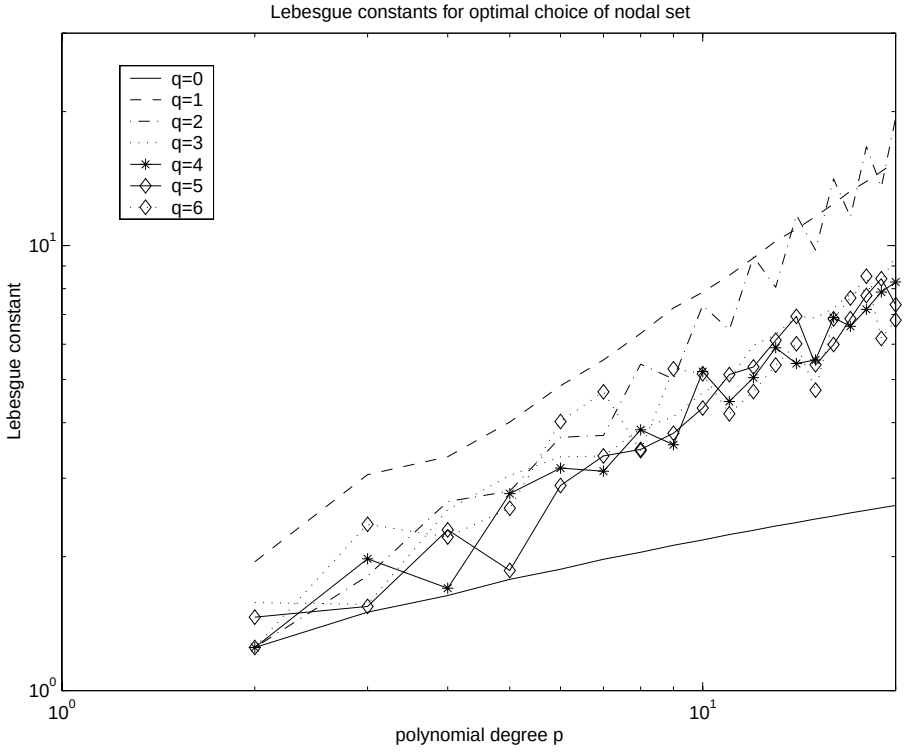

Fig. 4. Lebesgue number of set $\mathcal{N}_1^{p,q}$ as a function of $p = 2, \ldots, 20$ and degree of overintegration q.

Table 3
Index lists of $\mathcal{GL}^{p+q} \setminus \mathcal{N}^{p,q}_{1,\mathrm{sym}}$

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 4
Index lists of $\mathcal{GL}^{p+q} \setminus \mathcal{N}^{p,q}_{2,\mathrm{sym}}$

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

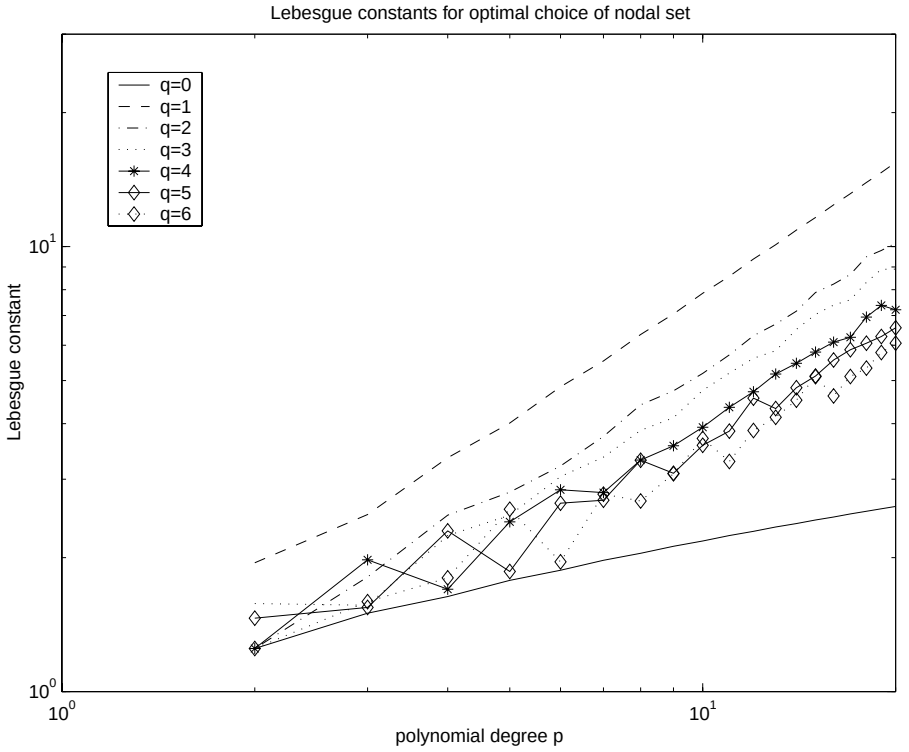

Fig. 5. Lebesgue number of set $\mathcal{N}_2^{p,q}$ as a function of $p = 2, \ldots, 20$ and degree of overintegration q.

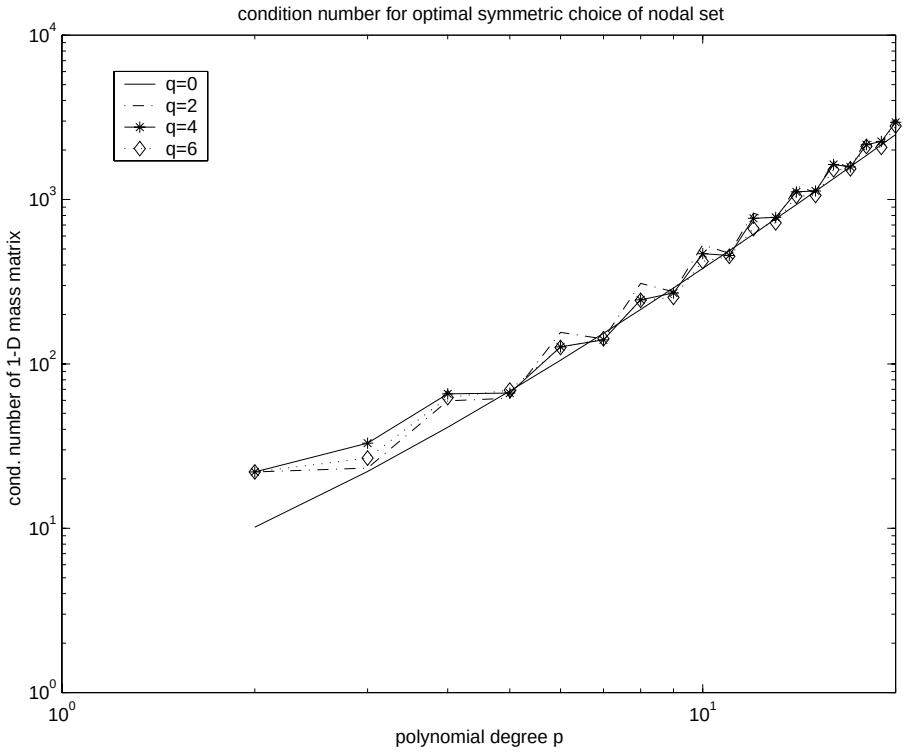

Fig. 6. $\mathrm{cond}\left(M^1(\mathcal{N}_{1,\mathrm{sym}}^{p,q})\right)$ as a function of $p = 2, \ldots, 20$ and degree of overintegration q.

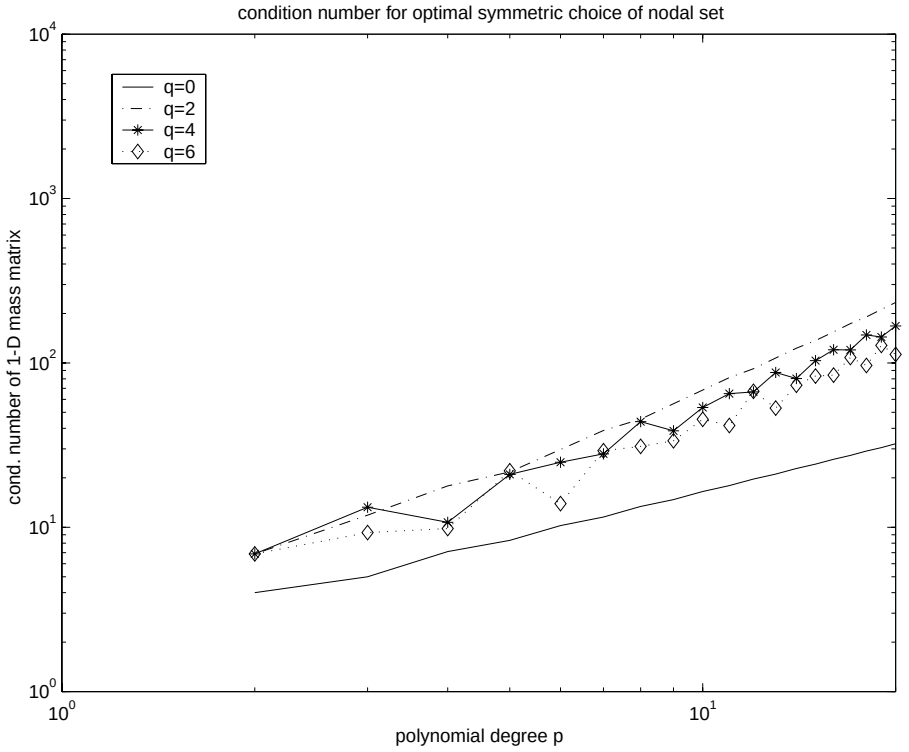

Fig. 7. $\mathrm{cond}\left(M^2(\mathcal{N}^{p,q}_{2,\mathrm{sym}})\right)$ as a function of $p = 2, \ldots, 20$ and degree of overintegration q.

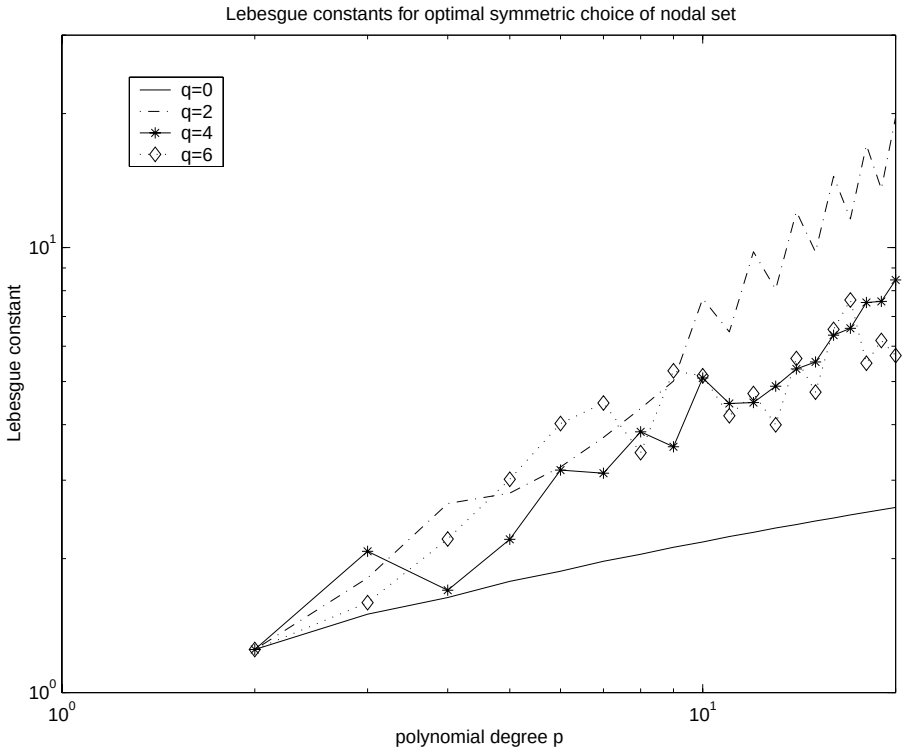

Fig. 8. Lebesgue number of set $\mathcal{N}^{p,q}_{1,\mathrm{sym}}$ as a function of $p = 2, \ldots, 20$ and degree of overintegration q.

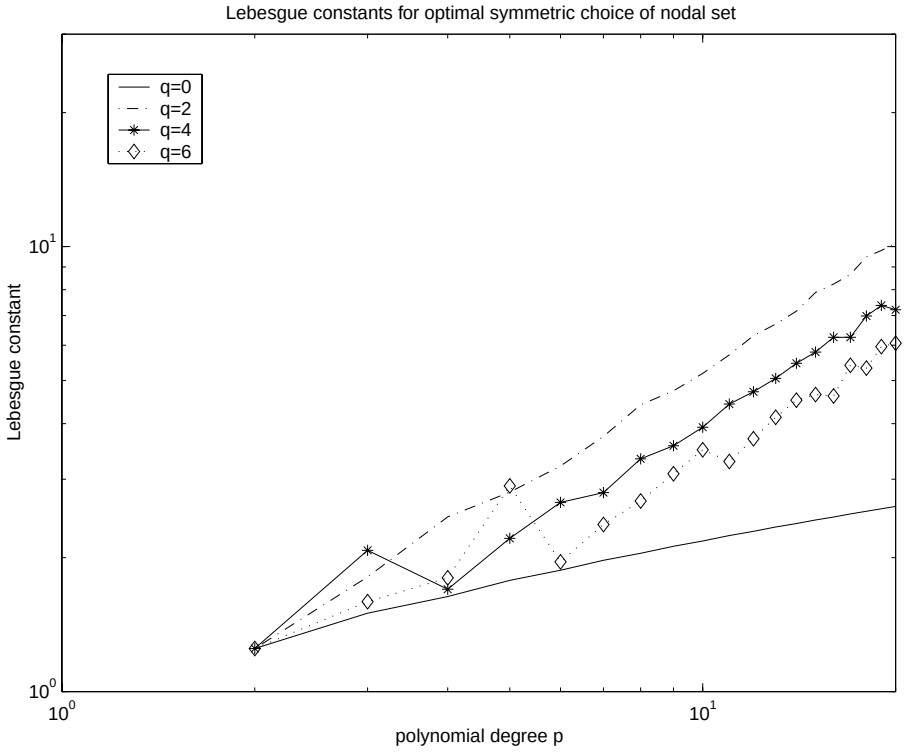

Fig. 9. Lebesgue number of set $\mathcal{N}^{p,q}_{2,\mathrm{sym}}$ as a function of $p = 2, \ldots, 20$ and degree of overintegration q.

Algorithm 2.13 (*Spectral Galerkin element algorithm*).

initialize $A_K = 0$
choose external shape functions $\mathscr{E}^k, k = 0, \ldots, 4$ for S_{p_K}
determine poly. deg. $p_1 := I^{\mathscr{I}} + 1$, $p_2 := J^{\mathscr{I}} + 1$ of internal shape functions
choose $q^h, q^v \geqslant 0$ 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 1–4
set $\mathscr{I} = \{l_i^{(p_1)}(x, \mathcal{N}^{p_1,q^h}) \cdot l_j^{(p_2)}(y, \mathcal{N}^{p_2,q^v}) | 1 \leqslant i \leqslant p_1 - 1, 1 \leqslant j \leqslant p_2 - 1\}$
for r, s = 1:2
  for $B_1 \in \{\mathscr{E}^0, \mathscr{E}^1, \mathscr{E}^2, \mathscr{E}^3, \mathscr{E}^4, \mathscr{I}\}$
    for $B_2 \in \{\mathscr{E}^0, \mathscr{E}^1, \mathscr{E}^2, \mathscr{E}^3, \mathscr{E}^4, \mathscr{I}\}$
      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

Remark 2.14. In Algorithm 2.13, 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 2.13, 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 several sets of shape functions that can be used in Algorithm 2.13. 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 [25] as

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

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 $K \in \mathscr{T}$ we introduce the element polynomial degree vector p_K as

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

Here, $p_K^i, 1 \leqslant i \leqslant 4$, represents the polynomial degree on the ith edge of the element, i.e., p_K^i is the approximation order associated with the midside node $\hat{a}_{i+4}$ in Fig. 10. $p_K^5 = (p_K^h, p_K^v)$ is the polynomial degree vector associated with the interior of the element; p_K^h and p_K^v represent the horizontal and vertical approximation orders, respectively. We mention in passing that the ability to use such anisotropic polynomial degree distribution for the internal degrees of freedom is essential in the context of thin solids. The nodes $\hat{a}_1, \cdots, \hat{a}_9$ represent the degrees of freedom corresponding to the quadrilateral master element shown in Fig. 10 and are divided into three categories:

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 $\mathscr{E}^0$; they are always taken as the set of the four bilinear functions:

$$\mathscr{E}^0 = \{h_i(x) \cdot h_j(y) | (i,j) \in \{1,2\} \times \{1,2\}\}. \tag{2.23}$$

2. Middle node $\hat{a}_9$: the shape functions associated with this node form the set $\mathscr{I}$; they are always taken in the form

$$\mathscr{I} = \left\{ l_i^{(p^h)}(x, \mathscr{N}^{p^h,q^h}) \cdot l_j^{(p^v)}(y, \mathscr{N}^{p^v,q^v}) \middle| 1 \leqslant i \leqslant p^h - 1, \quad 1 \leqslant j \leqslant p^v - 1 \right\}. \tag{2.24}$$

   Here, the sets $\mathscr{N}^{p^h,q^h}, \mathscr{N}^{p^v,q^v}$ can be any of those from Tables 1–4.
3. Midside 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 $\mathscr{E}^k, k = 1, \ldots, 4$; 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

$$\mathscr{E}^1 = \{h_i(x) \cdot h_1(y) | 1 \leqslant i \leqslant p_1 - 1\}, \tag{2.25a}$$

$$\mathscr{E}^2 = \{h_2(x) \cdot h_i(y) | 1 \leqslant i \leqslant p_2 - 1\}, \tag{2.25b}$$

$$\mathscr{E}^3 = \{h_i(x) \cdot h_2(y) | 1 \leqslant i \leqslant p_3 - 1\}, \tag{2.25c}$$

$$\mathscr{E}^4 = \{h_1(x) \cdot h_i(y) | 1 \leqslant i \leqslant p_4 - 1\}. \tag{2.25d}$$

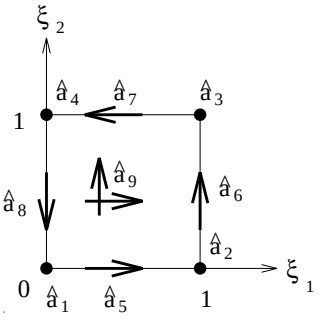

Fig. 10. Quadrilateral master element $\hat{K}$ of order $p = (p^1, p^2, p^3, p^4, p^5)$.

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

$$\mathscr{E}^1 = \left\{ l_i^{(p_1)}(x, \mathscr{GL}^{p_1}) \cdot h_1(y) | 1 \leqslant i \leqslant p_1 - 1 \right\}, \tag{2.26a}$$

$$\mathscr{E}^2 = \left\{ h_2(x) \cdot l_i^{(p_2)}(y, \mathscr{GL}^{p_2}) | 1 \leqslant i \leqslant p_2 - 1 \right\}, \tag{2.26b}$$

$$\mathscr{E}^3 = \left\{ l_i^{(p_3)}(x, \mathscr{GL}^{p_3}) \cdot h_2(y) | 1 \leqslant i \leqslant p_3 - 1 \right\}, \tag{2.26c}$$

$$\mathscr{E}^4 = \left\{ h_1(x) \cdot l_i^{(p_4)}(y, \mathscr{GL}^{p_4}) | 1 \leqslant i \leqslant p_4 - 1 \right\}. \tag{2.26d}$$

Remark 2.15. The nodal sets $\mathscr{GL}^{p_1}, \ldots,$ may be replaced by other sets, e.g., the sets $\mathscr{N}^{p,q}$. 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; note that Algorithm 2.10 will exploit this automatically.

2.4.2. Side shape functions of type (2.18)

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

$$\mathscr{E}^1 = \left\{ h_i(x) \cdot l_0^{(p^v)}(y, \mathscr{N}^{p^v,q^v}) | 1 \leqslant i \leqslant p_1 - 1 \right\}, \tag{2.27a}$$

$$\mathscr{E}^2 = \left\{ l_{p^h}^{(p^h)}(x, \mathscr{N}^{p^h,q^h}) \cdot h_i(y) | 1 \leqslant i \leqslant p_2 - 1 \right\}, \tag{2.27b}$$

$$\mathscr{E}^3 = \left\{ h_i(x) \cdot l_{p^v}^{(p^v)}(y, \mathscr{N}^{p^v,q^v}) | 1 \leqslant i \leqslant p_3 - 1 \right\}, \tag{2.27c}$$

$$\mathscr{E}^4 = \left\{ l_0^{(p^h)}(x, \mathscr{N}^{p^h,q^h}) \cdot h_i(y) | 1 \leqslant i \leqslant p_4 - 1 \right\}, \tag{2.27d}$$

where the sets $\mathscr{N}^{p^h,q^h}, \mathscr{N}^{p^v,q^v}$ 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; this fact is automatically exploited by Algorithm 2.10.

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

$$\mathscr{E}^1 = \left\{ l_i^{(p_1)}(x, \mathscr{GL}^{p_1}) l_0^{(p^v)}(y, \mathscr{N}^{p^v,q^v}) | 1 \leqslant i \leqslant p_1 - 1 \right\}, \tag{2.28a}$$

$$\mathscr{E}^2 = \left\{ l_{p^h}^{(p^h)}(x, \mathscr{N}^{p^h,q^h}) l_i^{(p_2)}(y, \mathscr{GL}^{p_2}) | 1 \leqslant i \leqslant p_2 - 1 \right\}, \tag{2.28b}$$

$$\mathscr{E}^3 = \left\{ l_i^{(p_3)}(x, \mathscr{GL}^{p_3}) l_{p^v}^{(p^v)}(y, \mathscr{N}^{p^v,q^v}) | 1 \leqslant i \leqslant p_3 - 1 \right\}, \tag{2.28c}$$

$$\mathscr{E}^4 = \left\{ l_0^{(p^h)}(x, \mathscr{N}^{p^h,q^h}) l_i^{(p_4)}(y, \mathscr{GL}^{p_4}) | 1 \leqslant i \leqslant p_4 - 1 \right\}. \tag{2.28d}$$

Again, as in the case (2.26a)–(2.26d), the sets $\mathscr{GL}^{p_1}, \ldots, \mathscr{GL}^{p_4}$ could be replaced with sets of the form $\mathscr{N}^{p,q}$.

Remark 2.16. 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 3-D

3.1. Work estimates in 3-D

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

Bases of the spaces S_{p_K} are commonly decomposed 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 that may be represented in the following fashion (note that a 3-D hexahedron has 8 vertices, 12 edges, and 6 faces for a total of 26 geometric entities)

$$\mathscr{E}^k = \left\{ \chi^m_{1,i}(\xi_1)\chi^m_{2,j}(\xi_2)\chi^m_{3,k}(\xi_3) | 1 \leqslant i \leqslant I_m, 1 \leqslant j \leqslant J_m, 1 \leqslant k \leqslant K_m \right\}, \quad m = 1, \ldots, 26,$$
$$\mathscr{I} = \left\{ \chi^{\mathscr{I}}_{1,i}(\xi_1)\chi^{\mathscr{I}}_{2,j}(\xi_2)\chi^{\mathscr{I}}_{3,k}(\xi_3) | 1 \leqslant i \leqslant I_{\mathscr{I}}, 1 \leqslant j \leqslant J_{\mathscr{I}}, 1 \leqslant k \leqslant K_{\mathscr{I}} \right\}.$$

Second, the numbers I_m, J_m, K_m satisfy

$$\forall m \in \{1, \ldots, 26\} \quad \text{at least one of the numbers } I_m, J_m, K_m \text{ is equal to 1.} \tag{3.29}$$

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 a typical decomposition of the space $Q_p(\hat{K})$, the values I_m, J_m, K_m, and $I_{\mathscr{I}}, J_{\mathscr{I}}, K_{\mathscr{I}}$ satisfy

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

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 $\mathrm{O}(p^4(p+q)^3)$ (the use of $GL_{p+q,p+q,p+q}$ is assumed). For fixed $q \geqslant 0$, the sum factorization Algorithm 2.5 thus generates the stiffness matrix with work $W = \mathrm{O}(p^7)$. Recall that, while Algorithm 2.5 is based on a fixed quadrature summation order, Algorithm 2.13 re-orders the quadrature summation order so as to minimize the total computational cost. We will now show that in 3-D, this re-ordering is essential for the efficiency of our *hp*-spectral Galerkin algorithm. In the 3-D version of Algorithm 2.10, six possible re-orderings of the triple quadrature sum have to be checked. However, in order to obtain an upper bound on the work for the 3-D version of Algorithm

Table 5
Asymptotic work estimates for various quadrature schemes with overintegration of degree $q \geqslant 0$

Algorithm	2-D	3-D
Standard	$\mathrm{O}(p^4(p+q)^2)$	$\mathrm{O}(p^6(p+q)^3)$
Sum factorization	$\mathrm{O}(p^4(p+q)+p^2(p+q)^2)$	$\mathrm{O}(p^6(p+q)+p^4(p+q)^2+p^2(p+q)^3)$
Spectral Galerkin	$\mathrm{O}(p^4(1+q)+p^2q^2)$	$\mathrm{O}(qp^6+p^5+q^2p^4+q^3p^2)$

2.10, it suffices to consider the operation count for one particular, judiciously chosen ordering. For definiteness' sake in the examples below, we assume that the "face blocks" $\mathscr{E}^{21}, \mathscr{E}^{22}, \mathscr{E}^{23}$ satisfy

$$(I_{21}, J_{21}, K_{21}) = (p-1, p-1, 1), \quad (I_{22}, J_{22}, K_{22}) = (p-1, 1, p-1), \quad (I_{23}, J_{23}, K_{23}) = (1, p-1, p-1).$$

Let us start with the complexity analysis of the computation of the $\mathscr{E}^m$–$\mathscr{E}^n$ blocks. As a typical case, we consider the call to `spec_sum_fact`$(\partial/\partial\xi_1)\mathscr{E}^{22}, (\partial/\partial\xi_2)\mathscr{E}^{23}, a)$. In order to obtain an upper bound for the cost of this call, we evaluate the triple quadrature sum as

$$A_{ijk,i'j'k'} = \sum_{q_3=0}^{p+q} \chi^{22}_{3,k}(x_{3,q_3})\chi^{23}_{3,k'}(x_{3,q_3})H_1(q_3, j, j', i, i'),$$

where

$$H_1(q_3, j, j', i, i') = \sum_{q_2=0}^{p+q} \chi^{22}_{2,j}(x_{2,q_2})\left(\chi^{23}_{2,j'}\right)'(x_{2,q_2})H_2(q_2, q_3, i, i'),$$

and

$$H_2(q_2, q_3, i, i') = \sum_{q_1=0}^{p+q} \left(\chi^{22}_{1,i}\right)'(x_{1,q_1})\chi^{23}_{1,i'}(x_{1,q_1})a(x_{1,q_1}, x_{2,q_2}, x_{3,q_3})w_{1,q_1}w_{2,q_2}w_{3,q_3}.$$

As $I_{22} = p-1, I_{23} = 1$, the auxiliary field H_2 is generated with work $W = \mathrm{O}(p(p+q)^2(p+q))$. Next, as $J_{22} = 1, J_{23} = p-1$, the field H_1 is computed with work $W = \mathrm{O}(p^2(p+q)(p+q))$. The final computation of the entries of $A_{ijk,i'j'k'}$ is achieved with work $W = \mathrm{O}(p^4(p+q))$. The total cost is therefore $W = \mathrm{O}(p^4(p+q) + p^2(p+q)^2 + p(p+q)^3)$. The reader may convince himself that this work estimate is indeed an upper bound for the work to compute each $\mathscr{E}^m$–$\mathscr{E}^n$ block. We note that for fixed $q \geqslant 0$, the complexity to compute the $\mathscr{E}^m$–$\mathscr{E}^n$ block is $\mathrm{O}(p^5)$. The re-ordering of the quadrature summation order in Algorithm 2.10 is essential for this $\mathrm{O}(p^5)$ estimate: If the summation order is fixed (e.g., as $\sum_{q_1}\sum_{q_2}\sum_{q_3}$) as in the classical sum factorization algorithm, then calling `sum_fact`$((\partial/\partial\xi_1)\mathscr{E}^{23}, (\partial/\partial\xi_2)\mathscr{E}^{23}, a)$ costs $W = \mathrm{O}(p^4(p+q) + p^4(p+q)^2 + p^2(p+q)^3)$, which, for fixed q, is of complexity $\mathrm{O}(p^6)$.

We now turn to the work estimates for the $\mathscr{I}$–$\mathscr{E}$ and $\mathscr{I}$–$\mathscr{I}$ blocks with internal shape functions that are adapted to the quadrature rule. As an example, we derive upper bounds for the work required to compute `spec_sum_fact`$((\partial/\partial\xi_1)\mathscr{I}, (\partial/\partial\xi_2)\mathscr{E}^{21}, a)$. We evaluate the triple quadrature sum as

$$A_{ijk,i'j'k'} = \sum_{q_2} \chi^{\mathscr{I}}_{2,j}(x_{2,q_2})\left(\chi^{21}_{2,j'}\right)'(x_{2,q_2})H_1(q_2, k, k', i, i'),$$

where

$$H_1(q_2, k, k', i, i') = \sum_{q_3} \chi^{\mathscr{I}}_{3,k}(x_{3,q_3})\chi^{21}_{3,k'}(x_{3,q_3})H_2(q_2, q_3, i, i'),$$

and

$$H_2(q_2, q_3, i, i') = \sum_{q_1} \left(\chi^{\mathscr{I}}_{1,i}\right)'(x_{1,q_1})\chi^{21}_{1,i'}(x_{1,q_1})a(x_{1,q_1}, x_{2,q_2}, x_{3,q_3})w_{1,q_1}w_{2,q_2}w_{3,q_3}.$$

Generating the auxiliary array H_2 can be achieved with work $W(H_2) = \mathrm{O}((p+q)^2p^2(p+q))$. 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 $W(H_1) = \mathrm{O}((p+q)p^3(1+q))$ 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 $W = \mathrm{O}(p^3p^2(1+q))$. The total cost is therefore $W(A) = \mathrm{O}(p^5(1+q) + q^2p^3 + q^3p^2)$. 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 $\mathscr{I}$–$\mathscr{E}$ blocks.

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

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

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

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

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

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 $\mathrm{O}(p^5)$. [2] 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 $\mathscr{I}$–$\mathscr{I}$ block of the stiffness matrix as only $\mathrm{O}(p^5)$ entries are non-zero of this block with $\mathrm{O}(p^6)$ elements.

We collect the work estimates for the standard Algorithm 2.2, the sum factorization Algorithm 2.5, and our Algorithm 2.13 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 2.13, 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 3-D: numerical example

In this section, we illustrate numerically the quadrature speed-up of Algorithm 2.13 over both the standard quadrature Algorithm 2.2 and the sum factorization Algorithm 2.5. All three algorithms were used to generate the stiffness matrix for a 3-D 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 3-D analogs of (2.23) for the vertex shape functions, of (2.24) for the internal shape functions, and of ((2.25a)–(2.25d)) for the edges and faces. The stiffness matrix calculation was done with the 3-D version of the *hp*-code HP90 [9], a general *hp*-code written in Fortran 90; actual calculations were performed on a SUN UltraSparc running at 336 MHz using the manufacturer provided F90 compiler with optimization flag (“-fast”) set. In Fig. 11, we compare three algorithms: the standard

[2] The work estimates in Table 5 ignore the initialization phase $A_K = 0$, which is also an $\mathrm{O}(p^6)$ algorithm. To obtain a truly $\mathrm{O}(p^5)$ 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 its $\mathrm{O}(p^6)$ contribution is negligible in the practical regime of polynomial degrees p.

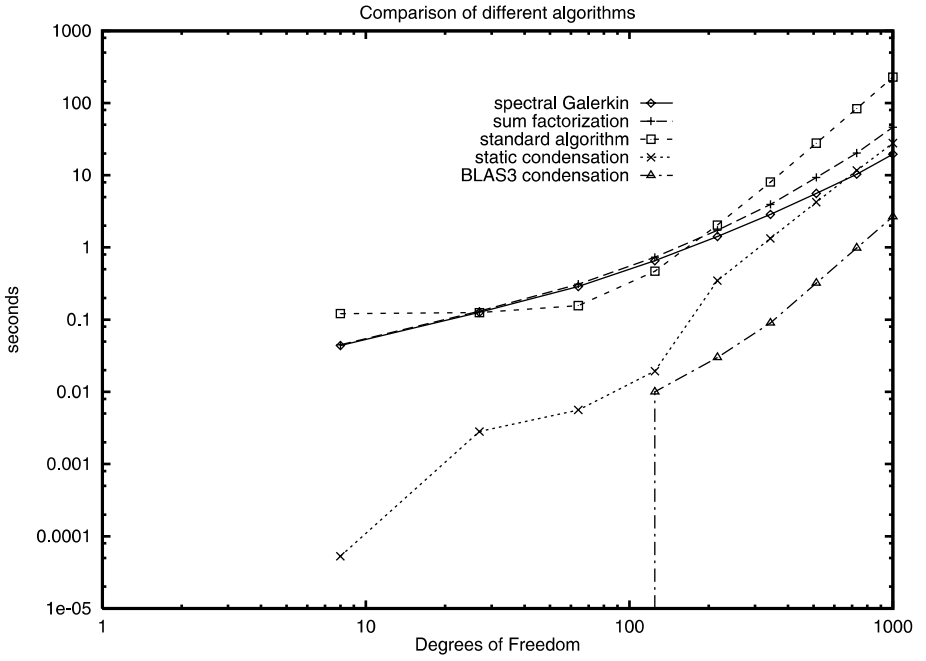

Fig. 11. CPU time per hexahedral element for standard, sum factorization, and spectral Galerkin algorithm for scalar 3-D model problem; also included: standard condensation and condensation based on optimized LAPACK and BLAS routines.

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 2.13 with Gauss–Lobatto quadrature rule $GL_{p,p,p}$ (i.e., $q=0$). As static condensation on the element level is often performed as part of the element generation, Fig. 11 includes the time required for this process. In fact, Fig. 11 contains the timings for two different condensation schemes: the faster of the two (labeled "blas3 condensation") uses the manufacturer-optimized LAPACK routine `dposv` and the BLAS3 routine `dgemm` while the slower one uses non-optimized routines. The spectral Galerkin algorithm outperforms both the sum factorization and the standard algorithm. For $p=9$, the stiffness matrix is generated in 228 s by the standard algorithm, in 46 s by the sum factorization, and in only 19.6 s by the spectral Galerkin algorithm. The regular static condensation takes 28 s, whereas the optimized one takes 2.7 s. Thus, for our scalar model problem and the range $p=1,\ldots,9$, our spectral Galerkin algorithm speeds up the element stiffness matrix generation significantly: the spectral Galerkin algorithm generates the condensed element matrix twice as fast as (even our improved version of) the sum factorization algorithm ($46+2.7$ s versus $19.6+2.7$ s).

Our analysis above shows that the order of complexity of the spectral Galerkin algorithm is lower than that of the sum factorization algorithm (cf. Table 5). Comparing the slopes of the corresponding curves in Fig. 11, we observe this reduced order of complexity for moderate values of p already.

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 and 13. Adapting the internal shape functions to the quadrature rule has the strongest impact on the timings for the $\mathscr{I}$–$\mathscr{I}$ block (the "bubble–bubble" block): While the calculation of the $\mathscr{I}$–$\mathscr{I}$ block is the most expensive part for the sum factorization algorithm (cf. Fig. 12), it is the cheapest one 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 the 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 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 $\mathscr{E}^n$–$\mathscr{E}^m$ blocks, the timings for the blocks not involving the internal shape functions are identical.

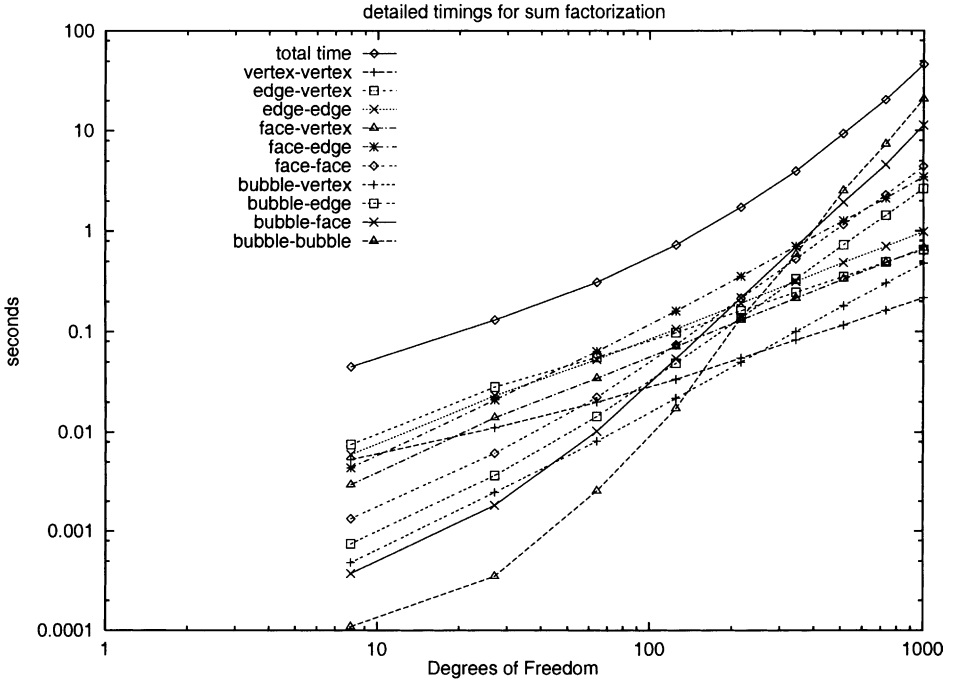

Fig. 12. Breakdown of timing for sum factorization algorithm for scalar 3-D model problem.

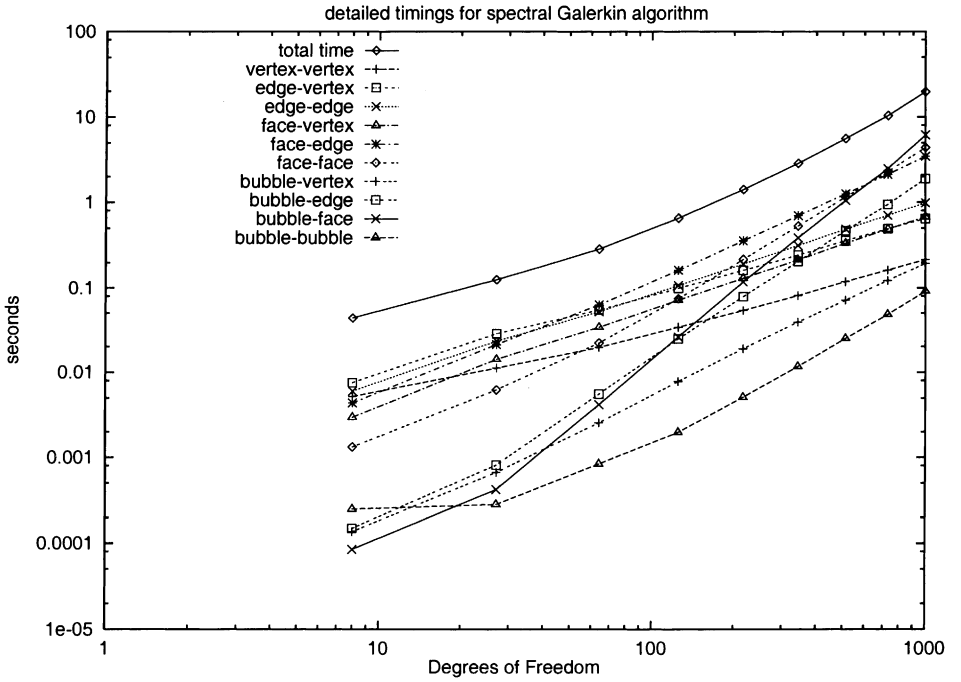

Fig. 13. Breakdown of timing for spectral Galerkin algorithm for scalar 3-D model problem.

Fig. 14 finally shows the timings for various auxiliary and book-keeping procedures. Evaluating the 1-D shape functions at the quadrature points, handling constrained nodes, and computing the right-hand side are seen to be negligible compared with the quadrature.

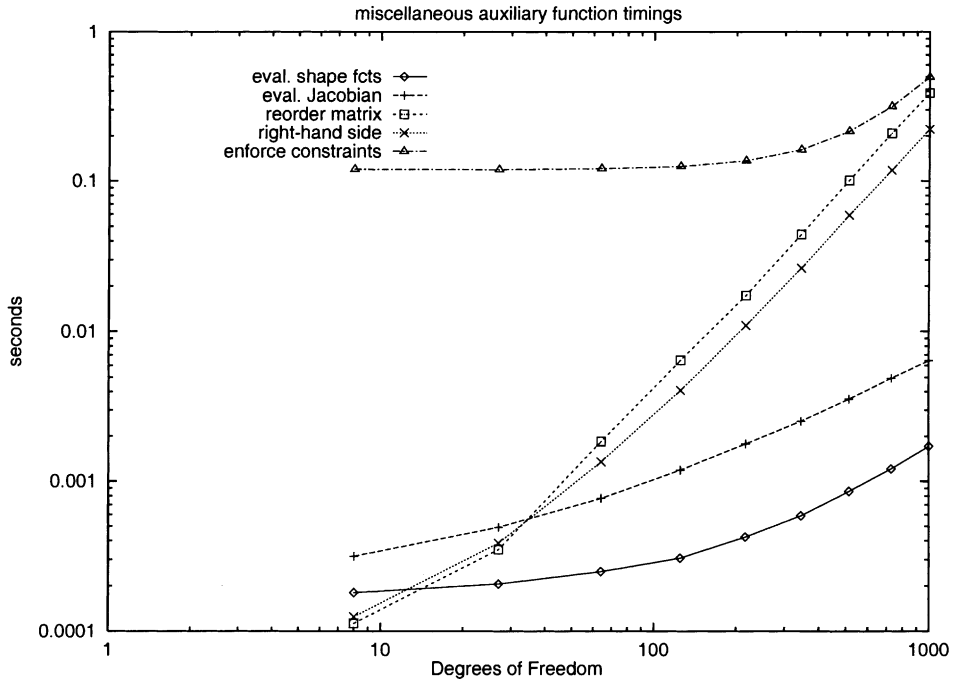

Fig. 14. Timings for auxiliary functions sum factorization/spectral Galerkin algorithm for scalar 3-D model problem.

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 a simple function of the spatial variables, 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 labeled "eval. Jacobian". We point out that this part of the computation depends 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 for two- and three-dimensional elliptic problems. The main idea consisted in employing sum factorization and in adapting the shape functions to the quadrature rule. A special case is the spectral element method, which is well established in computational fluid dynamics. The spectral element method is well known, however, to underintegrate the element stiffness matrices. Underintegration is known to cause stability problems if applied to elliptic systems arising in solid mechanics. We therefore generalized the ideas of the spectral element method by decoupling the quadrature order from the elemental degree, as is customary in Galerkin *hp*-FEM. Lagrange shape functions 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 and will be presented elsewhere.

Acknowledgements

We thank Philipp Frauenfelder for implementing the 3-D quadrature rules in the code HP90 [9].

References

[1] M. Ainsworth, A preconditioner based on domain decomposition for *hp*-FE approximation on quasi-uniform meshes, SIAM J. Numer. Anal. 33 (1996) 1358–1376.
[2] R. Actis, B. Szabó, C. Schwab, Hierarchic models for laminated plates and shells, Comput. Methods Appl. Mech. Engrg. 172 (1999) 79–107.
[3] I. Babuška, M. Suri, The *p* and *hp* versions of the finite element methods, basic principles and properties, SIAM Rev. 36 (1994) 578–632.
[4] C. Bernardi, Y. Maday, in: Approximations spectrales de problèmes aux limites elliptiques, Springer, Berlin, 1992.
[5] F. Ben Belgacem, Y. Maday, A spectral element methodology tuned to parallel implementations, Comput. Methods Appl. Mech. Engrg. 116 (1994) 59–67.
[6] C. Canuto, M.Y. Hussaini, A. Quarteroni, T.A. Zhang, in: Spectral Methods in Fluid Dynamics, Springer, Berlin, 1986.
[7] L. Demkowicz, J.T. Oden, W. Rachowicz, O. Hardy, Toward a universal *hp* adaptive finite element strategy. Part 1: Constrained approximation and data structure, Comput. Methods Appl. Mech. Engrg. 77 (1989) 79–112.
[8] L. Demkowicz, W. Rachowicz, K. Banas, J. Kucwaj. 2-D *hp* Adaptive Package (2D*hp*AP), Technical Report, Section of Applied Mathematics, Technical University of Cracow, Poland, 1992.
[9] L. Demkowicz, K. Gerdes, C. Schwab, A. Bajer, T. Walsh, HP90: A general and flexible Fortran 90 *hp*-FE code, Comput. Visualization Sci. 1 (1998) 145–163.
[10] M.O. Deville, E.H. Mund, Finite-element preconditioning for pseudospectral solutions of elliptic problems, SIAM J. Sci. Stat. Comput. 11 (1990) 311–342.
[11] J.M. Melenk, C. Schwab, Fully Discrete *hp*-FEM: Error Analysis, in preparation.
[12] B. Guo, W. Cao, An additive Schwarz method for the *hp*-version of the finite element method in three dimensions, SIAM J. Numer. Anal. 35 (1998) 632–654.
[13] D. Gottlieb, S.A. Orszag, Numerical analysis of spectral methods: theory and applications, CBMS 26, SIAM, 1977.
[14] W. Hackbusch, in: Multigrid Methods and Applications, Springer, Berlin, 1985.
[15] G.E. Karniadakis, S.J. Sherwin, in: Spectral/*hp* element methods for CFD, Oxford University Press, Oxford, 1999.
[16] P. Le Tallec, 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.
[17] Y. Maday, A.T. Patera, Spectral element method for Navier–Stokes equations, in: A.K. Noor, J.T. Oden (Eds.), State of the Art Surveys in Computational Mechanics, 1989, pp. 71–143.
[18] J.T. Oden, A. Patra, Y. Feng, Parallel domain decomposition solver for adaptive *hp* finite element methods, SIAM J. Numer. Anal. 34 (1997) 2090–2118.
[19] S.A. Orszag, Spectral methods for problems in complex geometries, J. Comput. Phys. 37 (1980) 70–92.
[20] A.T. Patera, Spectral element method for fluid dynamics: Laminar flow in a channel expansion, J. Comput. Phys. 54 (1984) 488–568.
[21] L. Pavarino, O. Widlund, A polylogarithmic bound for an iterative substructuring method for spectral element methods in three dimensions, SIAM J. Numer. Anal. 33 (1996) 1303–1355.
[22] A. Quarteroni, E. Zampieri, Finite element preconditioning for Legendre spectral collocation approximations to elliptic equations and systems, SIAM J. Numer. Anal. 29 (1992) 917–936.
[23] C. Schwab, in: *p* and *hp* FEM, Oxford University Press, Oxford, 1998.
[24] Burkhard Sündermann, Lebesgue constants in Lagrangian interpolation at the Fekete points. Ergebnisberichte der Lehrstühle Mathematik III und VIII (Angewandte Mathematik) 44, Universität Dortmund, 1980.
[25] B. Szabó, I. Babuška, in: Finite Element Analysis, Wiley, New York, 1991.