Journal of Computational Physics 229 (2010) 5161–5181

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

From h to p efficiently: Implementing finite and spectral/hp element methods to achieve optimal performance for low- and high-order discretisations

Peter E.J. Vos [a,b], Spencer J. Sherwin [a,*], Robert M. Kirby [c,1]

[a] Department of Aeronautics, Imperial College London, London, UK
[b] Flemish Institute for Technological Research (Vito), Mol, Belgium
[c] School of Computing, University of Utah, Salt Lake City, UT, USA

A R T I C L E I N F O

Article history:
Received 24 November 2009
Received in revised form 22 March 2010
Accepted 23 March 2010
Available online 28 March 2010

Keywords:
Spectral/*hp* element method
Implementation
Elliptic problems

A B S T R A C T

The spectral/*hp* element method can be considered as bridging the gap between the – traditionally low-order – finite element method on one side and spectral methods on the other side. Consequently, a major challenge which arises in implementing the spectral/*hp* element methods is to design algorithms that perform efficiently for both low- and high-order spectral/*hp* discretisations, as well as discretisations in the intermediate regime. In this paper, we explain how the judicious use of different implementation strategies can be employed to achieve high efficiency across a wide range of polynomial orders. Furthermore, based upon this efficient implementation, we analyse which spectral/*hp* discretisation (which specific combination of mesh-size h and polynomial order P) minimises the computational cost to solve an elliptic problem up to a predefined level of accuracy. We investigate this question for a set of both smooth and non-smooth problems.

1. Introduction

The spectral/*hp* element method combines the geometric flexibility of classical *h*-type finite element or finite volume techniques with the desirable resolution properties of spectral methods. In this approach a polynomial expansion of order P is applied to every elemental domain of a coarse finite element type mesh. These techniques have been applied in many fundamental studies of fluid mechanics [1] and more recently have gained greater popularity in the modelling of wave-based phenomena such as computational electromagnetics [2] and shallow water problems [3] – particularly when applied within a Discontinuous Galerkin formulation.

Spectral/*hp* element methods can be considered as a high-order extension of classical – traditionally low-order – finite element methods where convergence is not only possible through reducing the characteristic mesh-size h but also through increasing the local polynomial order P within an elemental subdomain. However, the concept of high and low-order discretisations can mean very different things to different communities. For example, the seminal works by Zienkiewicz and Taylor [4] and Hughes [5] list examples of elemental expansions only up to third or possibly fourth-order, implying that these

* Corresponding author.
E-mail addresses: peter.vos@imperial.ac.uk (P.E.J. Vos), s.sherwin@imperial.ac.uk (S.J. Sherwin), kirby@cs.utah.edu (R.M. Kirby).
[1] Work accomplished while on sabbatical at Imperial College London.

0021-9991/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2010.03.031

orders are considered to be high-order for the traditional *h*-type finite element community. In contrast the text books of the spectral/*hp* element community [6–9] typically show examples of problems ranging from a low-order bound of minimally fourth-order up to anything ranging from 10th to 15th order polynomial expansions. On the other end of the spectrum, practitioners of global spectral methods [10] would probably consider a 16th order global expansion to be relatively low-order approximation.

One could wonder whether these different definitions of low- and high-order is just inherent to the tradition and lore of each of the communities or whether there are more practical reasons for this distinct interpretation. Proponents of lower order methods might highlight that some problems of practical interest are so geometrically complex that one cannot computationally afford to use high-order techniques on the massive meshes required to capture the geometry. Alternatively, proponents of high-order methods highlight that if the problem of interest can be captured on a computational domain at reasonable cost then using high-order approximations for sufficiently *smooth* solutions will provide a lower accuracy for a given computational cost. If one however probes even further it also becomes evident that the different communities choose to implement their algorithms in a different manner. For example the standard *h*-type finite element community will typically uses techniques such as sparse matrix storage formats (where only the non-zero entries of a global matrix are stored) to represent a global operator. In contrast the spectral/*hp* element community acknowledges that for higher polynomial expansions more closely coupled computational work takes place at the individual elemental level and this leads to the use of elemental operators rather than global matrix operators. In addition the global spectral method community often make use of the tensor-product approximations where products of one-dimensional rules for integration and differentiation can be applied. From the results in this paper, it will appear that each of the different implementation strategies will perform poorly when applied outside the aforementioned polynomial regimes typically adopted by the different communities, hinting that the traditional views of low- and high-order may be have been strengthened by these practical barriers.

In this paper, we are therefore lead to ask when we should adopt these different implementation strategies if we are to allow the order of our spatial approximations to vary from $P = 1$ up to say $P = 15$? We note that analytic estimates of computational work in this polynomial regime are difficult if not impossible to establish since the computational effort is highly dependent on hard to predict hardware characteristics such as memory management and caching effect as well as optimised linear algebra packages such as BLAS and LAPACK. We therefore will mainly follow a computational approach to assess the efficiency of the different implementation strategies. The support of various implementation strategies within a spectral/*hp* code will allow the user to cross the community dependent barriers of low- and high-order in an efficient way such as the aforementioned example of $P = 4$ (the high-order limit for traditional finite elements and the low-order limit for spectral/*hp* elements). This surrounding polynomial regime ($2 < P < 6$) is however potentially an optimal/desirable range for applications where the mesh resolution is such that increasing polynomial order leads to the onset of rapid/spectral convergence. This level of resolution might be necessary to capture, for example, a complex geometry. The benefit of intermediate polynomial resolution will however only be observed if one can efficiently implement these polynomial discretisations.

Finally, we can also question whether it is sufficient to know which implementation strategy is optimal in terms of CPU time for a specific polynomial order discretisation? Probably a more pertinent question is consider *given the most efficient implementation, what is the best spectral/hp discretisation to obtain a fixed error for a minimal computational cost?* Since computational cost is impacted by different discretisation methodologies such as element size *h*, polynomial order *P*, adaptive refinement *r* [11] or even the continuity of the approximation *k* [12], there are clearly many factors to consider. To help reduce this extensive parameter space in this paper we will restrict ourselves to just *h* and *P* refinements leading to the question: which specific combination of mesh-size *h* and polynomial order *P* requires the minimal computational cost (*i.e.* run-time) to solve a problem up to a predefined accuracy? We will investigate this question for a set of both smooth and non-smooth elliptic problems. To outline the scope of this study, we must also consider which part of the implementation we look to optimise. In the solution of time-dependent partial differential equations, such as those that arise in fluid mechanics, electromagnetics and oceanography, it is often the case that you can have the repeated application of a matrix problem. However, in a steady partial differential equation which might arise in structural mechanics the cost of setting up and the matrix problem may be as equally important as the solution time. In the following analysis we will adopt the first case and assume the repeated application of the matrix operators is the dominant cost. As a result, we aim to optimise the evaluation of such matrix operators for minimal run-time, thereby neglecting any matrix construction time. In addition, we do not consider any memory constraints.

The paper is organised as follows: in Section 2, we begin by introducing the spectral/*hp* element method and highlighting some of its aspects relevant to the topic of this paper. In Section 3, we explain how a local operator can be efficiently evaluated for both low- and high-order expansions. Therefore, we first introduce and discuss three different implementation strategies using global matrices, local matrices of a sum factorisation technique. We then provide theoretical cost estimates for each strategy. Next we investigate the effect of the different strategies on the actual run-time by a set of computational tests and analyse which strategy should be selected depending on the polynomial order. Subsequently, in Section 4 we look for the optimal combination of mesh-size *h* and polynomial order *P* that solves a certain problem up to a predefined level of accuracy in a minimal amount of time for four different test-problems. Finally Section 5 summarises and concludes the presented work.

2. The spectral/*hp* element method

2.1. Tensorial expansion bases

Like any finite element method, the spectral/*hp* element method starts by decomposing the domain Ω into a tessellation of $|\mathcal{E}|$ quadrilateral and/or triangular elements such that

$$\Omega = \bigcup_{e\in\mathcal{E}} \Omega_e. \tag{1}$$

Each of these elements Ω_e in *physical space* can be considered as an image of a standard element Ω_{st} in *reference space*. For every element, there exists a one-to-one mapping relating the physical Cartesian coordinates (x_1,x_2) of the element Ω_e to the reference coordinate system (ξ_1,ξ_2), which is defined as

$$x_1 = \chi_1^e(\xi_1,\xi_2), \quad x_2 = \chi_2^e(\xi_1,\xi_2). \tag{2}$$

2.1.1. Quadrilateral tensorial expansion bases

The quadrilateral standard element Ω_{st} is defined as the bi-unit square $\mathcal{Q}^2 = \{(\xi_1,\xi_2) \in [-1,1]\times[-1,1]\}$. Its Cartesian structure can be exploited to locally represent the solution as a tensorial spectral/*hp* expansion

$$u(\xi_1,\xi_2) = \sum_{n\in\mathcal{N}} \phi_n(\xi_1,\xi_2)\hat{u}_n = \sum_{p=0}^{P}\sum_{q=0}^{P} \psi_p(\xi_1)\psi_q(\xi_2)\hat{u}_{pq}, \tag{3}$$

where the set of two-dimensional basis functions $\{\phi_n\}$ with index set $\mathcal{N}$ is defined as a tensor product of the one-dimensional basis functions $\{\psi_p\}$ in each of the coordinate directions, as depicted in Fig. 1. The expansion basis $\{\psi_p\}$ spans the polynomial space of order P and, for a globally C^0 continuous expansion – see also Section 2.2 – typically consists of a set of either *modal* or *nodal* basis functions which can be decomposed into *boundary modes* and *interior modes*. Boundary modes are defined as all the modes which have non-zero support on the boundary where interior modes are zero on all boundaries. The tensor-product nature of the expansion will be the necessary prerequisite for the application of the sum-factorisation technique outlined in Section 3.1.1.

2.1.2. Triangular tensorial expansion bases

The non-tensorial structure of the triangular reference element $\mathcal{T}^2 = \{-1 \leqslant \xi_1,\xi_2; \xi_1+\xi_2 \leqslant 0\}$ seems to prohibit the construction of a tensorial expansion basis and consequently, the application of the sum-factorisation technique. However, after introducing a *collapsed* coordinate system given by the transformation

$$\eta_1(\xi_1,\xi_2) = 2\frac{1+\xi_1}{1-\xi_2} - 1, \quad \eta_2(\xi_1,\xi_2) = \xi_2, \tag{4}$$

the triangular element can now be defined as $\mathcal{T}^2 = \{(\eta_1,\eta_2) \in [-1,1]\times[-1,1]\}$. In order to generate a C^0-continuous expansion and to ensure completeness of the expansion, we now use a *generalised* tensor product to define the expansion basis as

$$u(\xi_1,\xi_2) = \sum_{n\in\mathcal{N}} \phi_n(\xi_1,\xi_2)\hat{u}_n = \sum_{p=0}^{P}\sum_{q=0}^{f(p)} \psi_p(\eta_1(\xi_1,\xi_2))\psi_{pq}(\eta_2(\xi_1,\xi_2))\hat{u}_{pq}. \tag{5}$$

Note that the upper bound $f(p)$ of the index q now depends on the index p. This construction is graphically represented in Fig. 2 for the C^0 continuous modal triangular expansion introduced by Dubiner [13]. For an in-depth discussion of tensorial expansions for triangular elements, the reader is referred to [7].

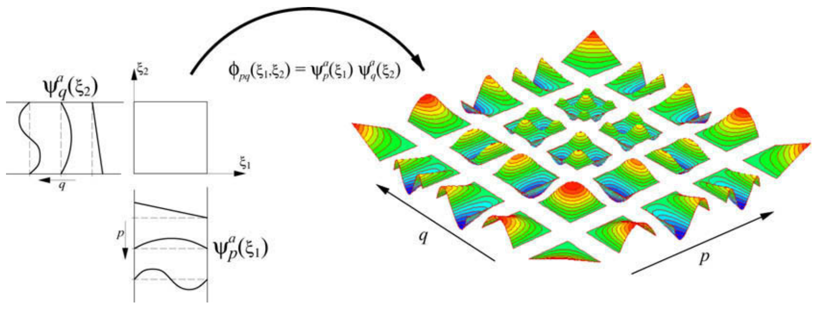
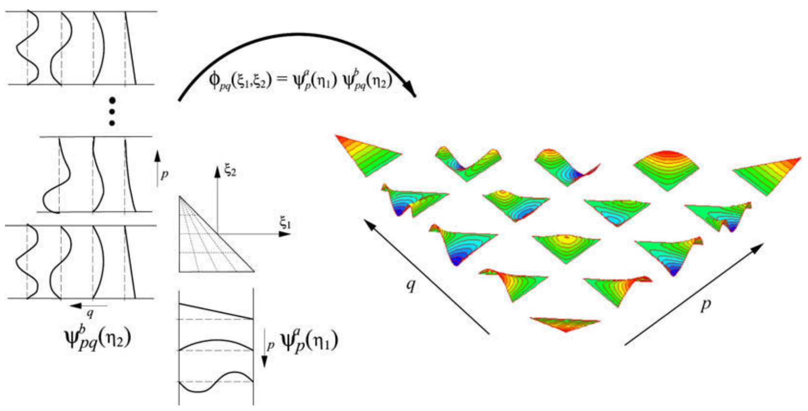


Fig. 1. Construction of a two-dimensional C^0 continuous modal quadrilateral expansion basis from the tensor product of two one-dimensional expansions of order $P = 4$ (edge and face modes have been scaled by a factor of 4 and 16, respectively).



$\phi_{pq}(\xi_1,\xi_2) = \psi^a_p(\eta_1)\,\psi^b_{pq}(\eta_2)$
p
q
ξ_2
ξ_1
$\psi^b_{pq}(\eta_2)$
$\psi^a_p(\eta_1)$

Fig. 2. Construction of a two-dimensional fourth-order C^0 continuous modal triangular expansion basis using a generalised tensor-product procedure (edge and face modes have been scaled by a factor of 4 and 16, respectively).

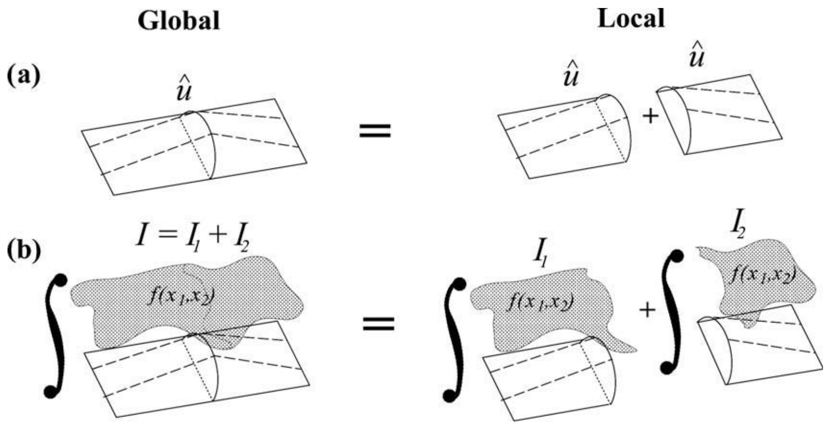


Fig. 3. Illustration of local to global assembly. If we have a global expansion as represented in figure (a) it can be decomposed into two elemental contributions multiplied by the same global coefficient $\hat{u}$. To integrate a function $f(x_1,x_2)$ with respect to the global mode, as illustrated in figure (b), the integration in the global region is the sum of the integration in the local regions.

Note that in both the quadrilateral and triangular case, the lowest order spectral/*hp* expansion (*i.e.* $P = 1$) corresponds to the well-known linear finite element expansion. The typical linear basis functions also appear as the vertex modes of higher order modal spectral/*hp* expansions, as can be observed in Figs. 1 and 2.

2.2. Global assembly

Although high-order expansion bases are initially constructed on an elemental level, we require some form of connectivity between the elements in order to solve partial differential equations. Such global representations of the solution are typically constructed by imposing C^0 continuity across element boundaries. This procedure is facilitated by the boundary/interior decomposition of the elemental modes as we consequently only need to merge the corresponding boundary modes of adjacent elements into single global modes. This is depicted in Fig. 3(a). A global C^0 continuous spectral/*hp* expansion can then be represented as

$$u(x_1,x_2) = \sum_{m\in\mathcal{N}^g} \Phi_m(x_1,x_2)\hat{u}^g_m = \sum_{e\in\mathcal{E}}\sum_{n\in\mathcal{N}} \phi^e_n(x_1,x_2)\hat{u}^e_n, \tag{6}$$

where $\{\hat{u}^g_i\}$ are the global degrees of freedom corresponding to the global expansion basis $\{\Phi_i\}$. The $|\mathcal{E}|\times|\mathcal{N}|$ elemental degrees of freedom $\hat{u}^e_m$ can be related to the $|\mathcal{N}^g|$ global degrees of freedom $\hat{u}^g_n$ through the local-to-global mapping $m(e,n)$. This mapping is embedded in the operator $\boldsymbol{\mathcal{A}}$ which scatters the vector of global coefficients $\hat{\boldsymbol{u}}_g$ upon the vector of local coefficients $\hat{\boldsymbol{u}}_l$ as

$$\hat{\boldsymbol{u}}_l = \boldsymbol{\mathcal{A}}\hat{\boldsymbol{u}}_g. \tag{7}$$

When following a traditional Galerkin procedure, the discrete weak formulation to be solved is an integral form which typically contains terms of the type

$$\boldsymbol{I}_g[m] := \int_{\Omega} \Phi_m(x_1,x_2) f(x_1,x_2) dx_1 dx_2, \quad m \in \mathcal{N}^g. \tag{8}$$

The elemental decomposition of the problem now allows us to express this inner product of the function $f(x_1,x_2)$ with respect to the global basis $\{\Phi_m\}$ as a series of elemental contributions such that

$$\boldsymbol{I}_g[m] := \sum \int_{\Omega^e} \phi^e_n(x_1,x_2) f(x_1,x_2) dx_1\, dx_2, \tag{9}$$

where the summation is taken over these elemental ϕ^e_n modes that correspond to the mth global mode as defined by the mapping $m(e,n)$. This process, referred to as *global assembly* or *direct stiffness summation* is graphically represented in Fig. 3(b) and can be mathematically expressed as the transpose of $\boldsymbol{\mathcal{A}}$ such that

$$\boldsymbol{I}_g = \boldsymbol{\mathcal{A}}^\top \boldsymbol{I}_l, \tag{10}$$

where $\boldsymbol{I}_l$ is the vector of local contributions. This procedure of global assembly can be used to for example construct the global mass matrix $\boldsymbol{M}$ as

$$\boldsymbol{M} = \boldsymbol{\mathcal{A}}^\top \underline{\boldsymbol{M}^e} \boldsymbol{\mathcal{A}}, \tag{11}$$

where $\underline{\boldsymbol{M}^e}$ is the block-diagonal concatenation of elemental matrices $\boldsymbol{M}^e$.

3. Evaluation of finite element operators

When following a standard Galerkin procedure, the weak form of a partial differential equations often comprise terms of the form

$$a(v,u), \tag{12}$$

where $a(\cdot,\cdot)$ is a bi-linear form and u,v are functions respectively belonging to a suitably chosen trial space $\mathcal{U}$ and test space $\mathcal{V}$. The discrete equivalent based upon the spectral/*hp* expansion (6) yields expressions of the form

$$\hat{y}^g_i := \sum_{j \in \mathcal{N}^g} a(\Phi_i, \Phi_j) \hat{u}^g_j, \tag{13}$$

which should typically be evaluated for all $i \in \mathcal{N}^g$. Using the matrix associated to this bi-linear form, this finite element operation can be written as

$$\hat{\boldsymbol{y}}_g = \boldsymbol{A}\hat{\boldsymbol{u}}_g, \tag{14}$$

where $\boldsymbol{A}[i][j] = a(\Phi_i, \Phi_j)$.

The goal of this section is to analyse how the operators of type (13) or (14) can be efficiently *evaluated* for both low- and high-order expansions. As a result, the focus is not on the matrix $\boldsymbol{A}$ itself, but merely on the action of $\boldsymbol{A}$. We seek for the most efficient way of mapping the vector $\hat{\boldsymbol{u}}_g$ to the vector $\hat{\boldsymbol{y}}_g$, without necessarily building $\boldsymbol{A}$ explicitly.

The action of finite element operators is required in various scenarios. Examples include:

- iterative solution techniques, which are founded on the forward operation of the matrix to be inverted,
- explicit time-stepping methods, which require an operator evaluation to compute the right-hand-side term of the semi-discrete system, and
- the strong enforcement of non-homogeneous Dirichlet (essential) boundary conditions (see also Section 4.1).

3.1. Evaluation strategies

In this section, we discuss three different strategies to evaluate expressions of the form (13). The first two strategies – referred to as the *sum-factorisation technique* and the *local matrix approach* – have in common that they both exploit the elemental decomposition of the spectral/*hp* element method. The contribution of each element is computed separately, whereafter the different contributions are assembled together using the direct stiffness summation technique as explained in Section 2.2. Both these approaches then evaluate (13) in the following three steps:

1. global-to-local mapping : $\hat{\boldsymbol{u}}_l = \mathcal{A}\hat{\boldsymbol{u}}_g,$ (15)
2. elemental evaluation : $\hat{y}^e_m = \sum_{n \in \mathcal{N}} a_e(\phi^e_m, \phi^e_n)\hat{u}^e_n \quad \forall m \in \mathcal{N}, e \in \mathcal{E},$ (16)
3. global assembly : $\hat{\boldsymbol{y}}_g = \mathcal{A}^\top \hat{\boldsymbol{y}}_l.$ (17)

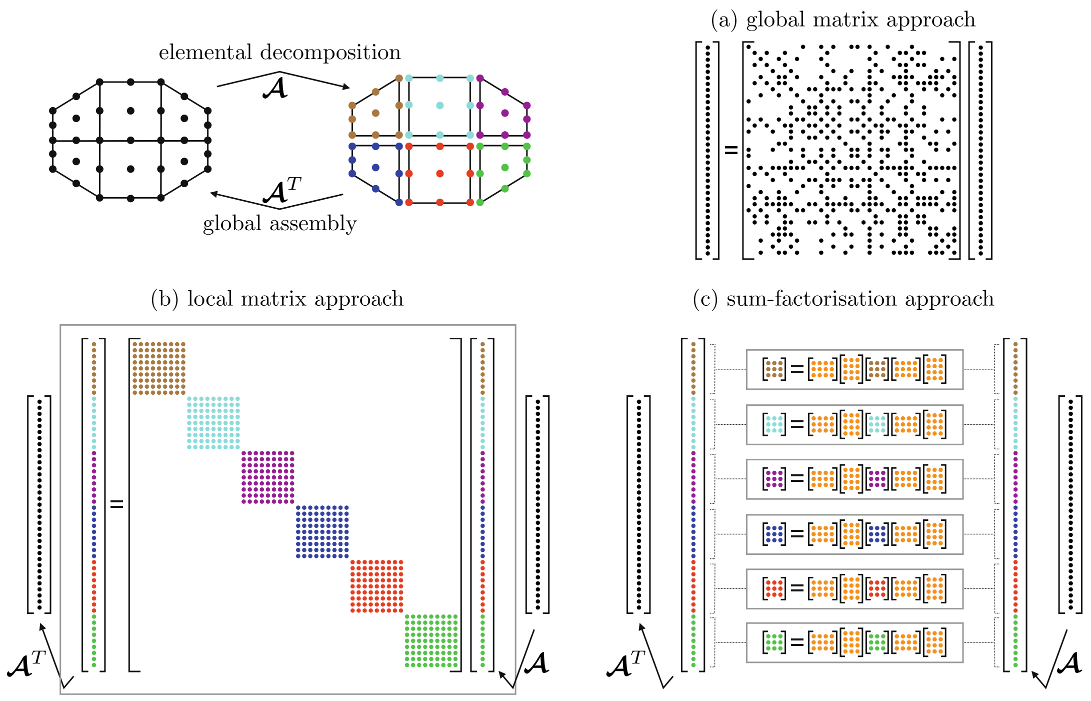

elemental decomposition
$\mathcal{A}$
$\mathcal{A}^T$
global assembly
(a) global matrix approach
(b) local matrix approach
(c) sum-factorisation approach

Fig. 4. A graphical representation of the different implementation strategies.

The question then becomes: how can the elemental form (16) be evaluated efficiently?

The third strategy using *global matrices*-differs from both these approaches in the sense that it is based on the global interpretation of the spectral/*hp* element method, rather than on its elemental decomposition.

A graphical representation of the different strategies is shown in Fig. 4. The existence of different evaluation strategies has been acknowledged before in [14,15]. In the present work, we will elaborate on a detailed cost analysis both from a theoretical (Section 3.2) as computational (Section 3.3) point of view and we particularly emphasise on the difference in results depending on the expansion order.

3.1.1. The sum-factorisation approach

To introduce the sum-factorisation technique, consider the example of the mass matrix operator. The elemental bi-linear form appearing in (16) is then defined as

$$a_e(\phi_m,\phi_n) = \int_{\Omega_e} \phi_m(\boldsymbol{x})\phi_n(\boldsymbol{x})d\boldsymbol{x}. \tag{18}$$

Making use of the coordinate transformation (2), this can be expressed as an integral over the reference domain Ω_{st}

$$a_e(\phi_m,\phi_n) = \int_{\Omega_{st}} \phi_m(\xi)\phi_n(\xi)|J(\xi)|d\xi, \tag{19}$$

where J is the Jacobian of the transformation.

The sum-factorisation technique is based upon two concepts: *numerical quadrature* and the use of tensorial basis functions.

3.1.1.1. Evaluation using numerical quadrature. Assume a quadrature rule of order Q to evaluate the integral

$$\int_{\Omega_{st}} f(\xi)d\xi = \sum_{r\in\mathcal{Q}} f(\xi_r)\omega_r. \tag{20}$$

Inserting (20) and (19) into (16) leads to the following expression to be evaluated:

$$\hat{y}_m = \sum_{n\in\mathcal{N}}\sum_{r\in\mathcal{Q}} \omega_r\phi_m(\xi_r)\phi_n(\xi_r)|J(\xi_r)|\hat{u}_n, \quad \forall m\in\mathcal{N}, \tag{21}$$

where for clarity, we have omitted the index e. After rearranging, this yields

$$\hat{y}_m = \sum_{r\in\mathcal{Q}} \phi_m(\xi_r)\omega_r|J(\xi_r)|\left\{\sum_{n\in\mathcal{N}}\phi_n(\xi_r)\hat{u}_n\right\}, \quad \forall m\in\mathcal{N}. \tag{22}$$

In matrix notation this simplifies to

$$\hat{\boldsymbol{y}} = \boldsymbol{B}^\top \boldsymbol{W}\boldsymbol{B}\hat{\boldsymbol{u}}, \tag{23}$$

where $\boldsymbol{B}[i][j] = \phi_j(\xi_i)$ is the discrete representation of the expansion basis and $\boldsymbol{W}$ is the diagonal matrix with entries $\boldsymbol{W}[i][i] = \omega_i|J(\xi_i)|$. For conciseness, we also have omitted the subscript l on the local coefficient vectors $\hat{\boldsymbol{y}}$ and $\hat{\boldsymbol{u}}$. It can be appreciated that the mass matrix operator can now be evaluated into two separate steps: a *backward transformation* $\boldsymbol{u} = \boldsymbol{B}\hat{\boldsymbol{u}}$ which evaluates the spectral/*hp* expansion at the quadrature points, and the *inner product operator* $\hat{\boldsymbol{y}} = \boldsymbol{B}^\top\boldsymbol{W}\boldsymbol{u}$ which subsequently takes the inner product of the function u with respect to all elemental expansion modes.

Remark 1. Despite the typical formulation (23) in matrix notation, the numerical quadrature approach can be classified as a *matrix-free* approach. It is matrix-free in the sense that evaluation of the operator is not dependent on the construction of the matrix associated to the bi-linear form.

3.1.1.2. Sum-factorisation for quadrilateral expansions. If both the spectral/*hp* expansion basis and the numerical quadrature rule exhibit a tensor-product structure, the backward transformation as well as the inner product can be factorised using the sum-factorisation technique. Consider the backward transformation $\boldsymbol{u} = \boldsymbol{B}\hat{\boldsymbol{u}}$ for a quadrilateral element, defined as

$$u(\xi_r) = \sum_{n\in\mathcal{N}} \phi_n(\xi_r)\hat{u}_n \quad \forall r \in \mathcal{Q}. \tag{24}$$

Acknowledging the tensorial nature of both the expansion and quadrature rule, this yields

$$u(\xi_{1s},\xi_{2t}) = \sum_{p=0}^{P}\sum_{q=0}^{P} \psi_p^a(\xi_{1s})\psi_q^a(\xi_{2t})\hat{u}_{pq} = \sum_{q=0}^{P}\psi_q^a(\xi_{2t})\left\{\sum_{p=0}^{P}\psi_p^a(\xi_{1s})\hat{u}_{pq}\right\}, \quad \forall s,t \in \{0,\dots,Q-1\}, \tag{25}$$

where we have *factored* the term $\psi_q^a(\xi_{2t})$ out of the second summation. In matrix notation, this is equivalent to:

$$\boldsymbol{u} = \boldsymbol{B}\hat{\boldsymbol{u}} = \left(\boldsymbol{B}_2^a \otimes \boldsymbol{B}_1^a\right)\hat{\boldsymbol{u}} = \left(\boldsymbol{B}_2^a \otimes \boldsymbol{I}\right)\left\{\left(\boldsymbol{I} \otimes \boldsymbol{B}_1^a\right)\hat{\boldsymbol{u}}\right\}, \tag{26}$$

where $\boldsymbol{B}_m^a[i][j] = \psi_j^a(\xi_i)$ represents the one-dimensional basis in direction m. Note that the relation $\boldsymbol{B} = \boldsymbol{B}_2^a \otimes \boldsymbol{B}_1^a$ reflects the tensorial structure of the expansion. Consequently, the backward transformation can then be evaluated into two separate steps:

A first step to compute the temporary variable $\boldsymbol{v} = (\boldsymbol{I} \otimes \boldsymbol{B}^a)\hat{\boldsymbol{u}}$:

$$v_q(\xi_{2s}) = \sum_{p=0}^{P}\psi_p^a(\xi_{1s})\hat{u}_{pq} \quad \forall q,s, \tag{27}$$

followed by the evaluation of $\boldsymbol{u} = (\boldsymbol{B}^a \otimes \boldsymbol{I})\boldsymbol{v}$:

$$u(\xi_{1s},\xi_{2t}) = \sum_{q=0}^{P}\psi_q^a(\xi_{2t})v_q(\xi_{2s}) \quad \forall s,t. \tag{28}$$

Furthermore, if we respectively consider $\hat{\boldsymbol{u}}$ and $\boldsymbol{u}$ as the column-major vectorisation of the matrices $\hat{\boldsymbol{U}}$ and $\boldsymbol{U}$, the notation can be further simplified to [8]

$$\boldsymbol{U} = \boldsymbol{B}_1^a\hat{\boldsymbol{U}}\boldsymbol{B}_2^{a\top}. \tag{29}$$

As a result, both sub-steps (27) and (28) can effectively be evaluated as the matrix–matrix multiplications $\boldsymbol{V} = \boldsymbol{B}_1^a\hat{\boldsymbol{U}}$ and $\boldsymbol{U} = \hat{\boldsymbol{V}}\boldsymbol{B}_2^{a\top}$, respectively. However, note that this does require a lexicographical ordering of the two-dimensional expansion modes $\hat{u}_{n(p,q)}$ with the p index running fastest. Also the vector $u_{r(s,t)}$ should follow a similar ordering along the first coordinate direction.

Analogously, the inner product operator $\hat{\boldsymbol{y}} = \boldsymbol{B}^\top\boldsymbol{W}\boldsymbol{u}$ can be factorised as

$$\hat{\boldsymbol{Y}} = \boldsymbol{B}_1^{a\top} w(\boldsymbol{U})\boldsymbol{B}_2^a, \tag{30}$$

where the function w multiplies each entry $\boldsymbol{U}[i][j] = u(\xi_{1i},\xi_{2j})$ with the corresponding quadrature metric $\omega_{i,j}|J(\xi_{1i},\xi_{2j})|$. Consequently, we can conclude that when adopting the sum-factorisation approach, the mass matrix operator can be evaluated as:

$$\hat{\boldsymbol{Y}} = \boldsymbol{B}^{a\top} w\left(\boldsymbol{B}^a\hat{\boldsymbol{U}}\boldsymbol{B}^{a\top}\right)\boldsymbol{B}^a. \tag{31}$$

This notation helps to appreciate that from an implementational point of view, this corresponds to a series of matrix–matrix products.

Remark 2. The use of numerical quadrature is not a necessary prerequisite for the application of the sum-factorisation technique. The evaluation of the elemental mass matrix operator can also be factorised as

$$\hat{w}_{m(p',q')} = |J| \sum_{p=0}^{P} \int_{-1}^{1} \psi_p^a(\xi_1)\psi_{p'}^a(\xi_1)d\xi_1 \left\{ \sum_{q=0}^{P} \int_{-1}^{1} \psi_q^a(\xi_2)\psi_{q'}^a(\xi_2)d\xi_2 \right\} \hat{u}_n \quad \forall p', q', \tag{32}$$

or equivalently,

$$\hat{\boldsymbol{w}} = |J| \left(\boldsymbol{M}^a \otimes \boldsymbol{M}^a \right) \hat{\boldsymbol{u}}, \tag{33}$$

where M^a is the mass matrix associated with the one-dimensional reference element. However, this approach implies that $|J|$ is a constant, which cannot be generally assumed. That is why in this work, the sum-factorisation technique will always be combined with the use of numerical quadrature.

3.1.1.3. Sum-factorisation for triangular expansions. For triangular elements, the use of a *generalised* tensor-product expansion, makes the sum-factorisation technique more complicated. While in Eq. (25), both the terms could be equally well being factorised out of the inner summation the backward transformation for triangles can only be factored as:

$$u(\xi_{1s}, \xi_{2t}) = \sum_{p=0}^{P} \sum_{q=0}^{f(p)} \psi_p^a(\xi_{1s})\psi_{pq}^b(\xi_{2t})\hat{u}_{pq} = \sum_{p=0}^{P} \psi_p^a(\xi_{1s}) \left\{ \sum_{q=0}^{f(p)} \psi_{pq}^b(\xi_{2t})\hat{u}_{pq} \right\} \quad \forall s, t. \tag{34}$$

In addition, both the summation bound $f(p)$ as the term ψ_{pq}^b of the inner summation now also depends on the index p, which prohibits a formulation similar to (26). Instead, the expression above can be written as

$$\boldsymbol{u} = \left(\boldsymbol{B}_1^a \otimes \boldsymbol{I} \right) \underline{\boldsymbol{B}}^b \hat{\boldsymbol{u}}, \tag{35}$$

where $\underline{\boldsymbol{B}}^b$ is the block-diagonal concatenation of the matrices $\boldsymbol{B}_p^b$ ($p = 0, \dots, P$) which are defined as $\boldsymbol{B}_p^b[i][j] = \psi_{pi}^b(\xi_{2j})$. As opposed to the quadrilateral case, this requires an ordering of the modes $\hat{u}_{n(p,q)}$ where q is running fastest. It can be appreciated that the first sub-step $\underline{\boldsymbol{B}}^b\hat{\boldsymbol{u}}$ cannot be implemented as a matrix–matrix product. Consequently, the implementation of (35) consist of $P + 1$ matrix–vector multiplications to evaluate $\boldsymbol{v} = \underline{\boldsymbol{B}}^b\hat{\boldsymbol{u}}$, followed by a matrix–matrix multiplication to evaluate $\boldsymbol{u} = (\boldsymbol{B}_1^a \otimes \boldsymbol{I})\boldsymbol{v}$ as $\boldsymbol{U} = \boldsymbol{B}_1^a\boldsymbol{V}^\top$. (In this last step we have chosen $\boldsymbol{U} = \boldsymbol{B}_1^a\boldsymbol{V}^\top$ over $\boldsymbol{U} = \boldsymbol{V}\boldsymbol{B}_1^{a\top}$ in order to ensure an ordering of $u_{r(i,j)}$ with the index i running fastest.)

The inner product and mass matrix operators for triangular expansions can be factorised in a similar way. This is discussed in more detail in [16].

3.1.2. The local matrix approach

The second elemental evaluation strategy is the *local matrix* approach. In this approach, the bi-linear form (16) is evaluated using the matrix associated to it. The evaluation of the elemental mass matrix operator then reduces to the matrix–vector multiplication

$$\hat{\boldsymbol{y}} = \boldsymbol{M}^e\hat{\boldsymbol{u}}, \tag{36}$$

where $\boldsymbol{M}^e[i][j] = \int_{\Omega_e} \phi_i^e \phi_j^e \, d\Omega_e$ is the elemental mass matrix. For the scope of this work, we do not consider the construction of the mass matrix to be part of the evaluation. We will assume that $\boldsymbol{M}^e$ has been precomputed (*e.g.* by means of (23) or (31)) and is readily available to evaluate (36).

3.1.3. The global matrix approach

As opposed to both the previous evaluation strategies, the global matrix approach acts directly upon the *global* bi-linear form (13). For the mass matrix operator, this form is then evaluated using the global mass matrix $\boldsymbol{M}$ as

$$\hat{\boldsymbol{y}}_g = \boldsymbol{M}\hat{\boldsymbol{u}}_g, \tag{37}$$

where $\boldsymbol{M}[i][j] = \int_{\Omega} \Phi_i \Phi_j \, d\Omega$. The finite element method typically leads to global matrices which are very *sparse*. For efficient storage and manipulation of sparse matrices, different formats only storing the non-zero entries of $\boldsymbol{M}$ have been proposed. Again, we assume that the global matrix $\boldsymbol{M}$ has been precomputed (*e.g.* through the global assembly procedure (11)) such that the global matrix evaluation only consist of the (sparse) matrix–vector multiplication.

3.2. Theoretical cost estimates based on operation count

In order to assess the efficiency of the different strategies, we will first analyse the operation count associated to each approach by investigating how many floating point operations each evaluation strategy requires.

3.2.1. Sum-factorisation versus local matrices

When comparing both the elemental strategies for evaluating the quadrilateral mass matrix, it can be observed that the factorised evaluation replaces the matrix–vector multiplication (36) by a series matrix–matrix multiplications, *i.e.* Eq. (31). However, while in the local matrix approach the single matrix $\boldsymbol{M}^e$ is of size $\mathcal{O}(P^2) \times \mathcal{O}(P^2)$, the matrices $\boldsymbol{B}^a_m$ in the sum-factorisation approach are only of size $\mathcal{O}(P) \times \mathcal{O}(P)$. This can also be explained as follows: the elemental mass matrix $\boldsymbol{M}^e$ truly is a two-dimensional operator while on the other hand, the sum-factorisation technique exploits the tensorial nature of expansion by applying the one-dimensional operators $\boldsymbol{B}^a_m$ along all the lines of constant ξ_n. As a result, the local matrix approach requires $\mathcal{O}(P^4)$ floating point operations to evaluate while the factorised evaluation only involves $\mathcal{O}(P^3)$ operations per matrix–matrix multiplication. This effectively is the strength of the sum-factorisation approach: it replaces an $\mathcal{O}(P^4)$ operation by multiple $\mathcal{O}(P^3)$ operations. This ensures a substantial performance benefit for the limit $P \to \infty$. However, for low-order expansions the coefficients of the leading order terms in the operation count do play an important role such that an exact operation count is required to assess whether the sum-factorisation technique truly reduces the number of floating point operations. A complete operation count analysis for four different finite element operators can be found back in [16], and the results are summarised in Table 1 and Figs. 5 and 6. From these data, it appears that for the backward transformation and the inner product operator – which for the factorised evaluation only involve two matrix–matrix multiplications – the sum-factorisation technique is always more efficient, except for the linear finite element case ($P = 1$) on triangular elements. However, for more complex operators such as the mass matrix operator or the Helmholtz operator defined as

$$a_e(\phi_n, \phi_m) = \int_{\Omega_e} \nabla\phi_n(\boldsymbol{x}) \cdot \nabla\phi_m(\boldsymbol{x}) - \lambda\phi_n(\boldsymbol{x})\phi_m(\boldsymbol{x})d\boldsymbol{x}, \tag{38}$$

the factorised evaluation, respectively, requires four and eight matrix–matrix multiplications. Consequently, the sum-factorisation technique only becomes more efficient from respectively $P = 5$ and $P = 10$ in the quadrilateral case. For triangular expansions, the break-even point may even be as high as $P = 27$ for the Helmholtz operator.

From the same data, it can also be observed for all operators that the sum-factorisation technique leads to a smaller reduction for triangular than for quadrilateral elements, both in terms of the break-even point as the reduction factor. This reduced effectiveness can be attributed to the triangular tensorial expansion not being constructed as a full tensor product.

Since Orszag's work [17] in 1980, the sum-factorisation technique has always been considered the key to the efficient implementation of global spectral methods (where a polynomial order $P = 100$ is deemed normal). However, the analysis in this section indicates that this cannot be considered as generally valid for the spectral element method in our chosen operational regime of $1 \leqslant P \leqslant 15$. The results show that sum-factorisation does not necessarily lead to a reduction in floating point operations, especially when evaluating complex operators for low-order expansions.

3.2.2. Global matrices versus local matrices

3.2.2.1. Quadrilateral expansions. For a multi-elemental spectral/*hp* expansion, it can be understood from [16] that the local matrix evaluation will require $2|\mathcal{E}|(P+1)^4$ floating point operations. On the other hand when adopting the global matrix strategy together with a sparse storage format to store $\boldsymbol{M}$, the operation count will scale like *nnz*, where *nnz* is the number of non-zero entries in $\boldsymbol{M}$. An entry $\boldsymbol{M}[i][j]$ is typically non-zero if the global basis function Φ_i and Φ_j are coupled, *i.e.* they have overlapping support. Estimates for the operation count are derived in [16] and the results are shown in Table 2 and Fig. 5.

The global matrix evaluation appears to be attractive in particular for low-order elements. In case $P = 1$ every global mode of a structured quadrilateral mesh is coupled to its nine neighbouring global modes (including itself), leading to nine multiply–add pairs per global DOF. However, every global mode corresponds to four local modes which each are coupled to the four linear elemental modes. As a result, the local matrix evaluation then requires $4 \times 4 = 16$ multiply–add pairs per global DOF. This implies that for $P = 1$ using global matrices only requires a fraction $9/16 = 0.5625$ of the floating point operations needed for the local matrix approach (assuming a sufficient mesh-size).

When increasing *P*, relatively more interior modes than edge modes will be added to each element. As there exist a one-to-one mapping between elemental interior modes and global expansion modes, the effect of the multiplicity of the elemen-

Table 1
Theoretical operation count (floating point multiplications and additions) for the elemental evaluation strategies.

		Sum-factorisation	Local matrix	Break-even
Quad	Backward transformation	$4P^3 + 18P^2 + 26P + 12$	$2P^4 + 12P^3 + 26P^2 + 24P + 8$	$P = 1$
	Inner product	$4P^3 + 19P^2 + 30P + 16$	$2P^4 + 12P^3 + 26P^2 + 24P + 8$	$P = 1$
	Mass matrix	$8P^3 + 37P^2 + 56P + 28$	$2P^4 + 8P^3 + 12P^2 + 8P + 2$	$P = 5$
	Helmholtz	$16P^3 + 93P^2 + 184P + 124$	$2P^4 + 8P^3 + 12P^2 + 8P + 2$	$P = 10$
Tri	Backward transformation	$3P^3 + 12P^2 + 17P + 8$	$P^4 + 6P^3 + 13P^2 + 12P + 4$	$P = 2$
	Inner product	$3P^3 + 13P^2 + 20P + 10$	$P^4 + 6P^3 + 13P^2 + 12P + 4$	$P = 2$
	Mass matrix	$6P^3 + 25P^2 + 37P + 18$	$0.5P^4 + 3P^3 + 6.5P^2 + 6P + 2$	$P = 11$
	Helmholtz	$14P^3 + 69P^2 + 113P + 58$	$0.5P^4 + 3P^3 + 6.5P^2 + 6P + 2$	$P = 27$

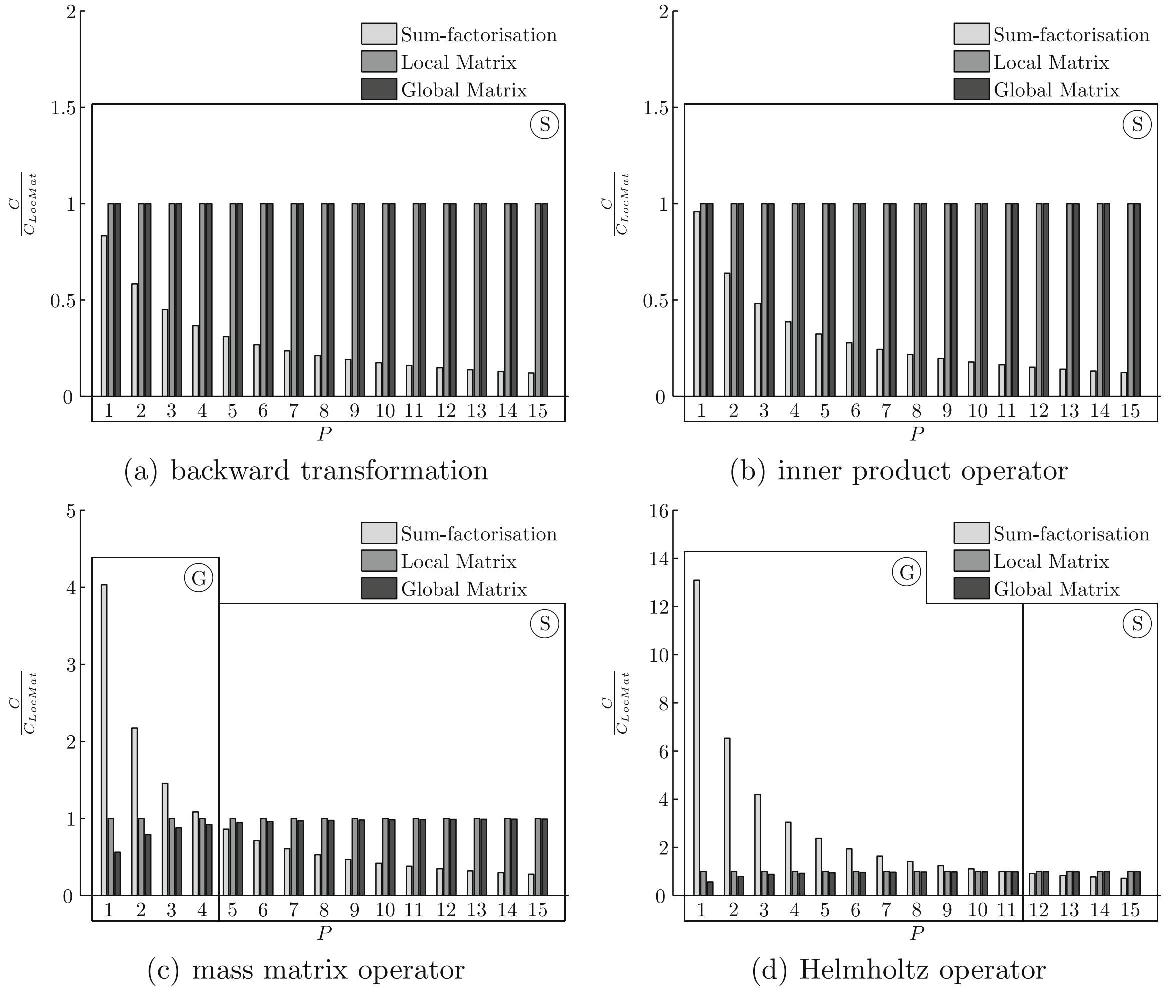


Fig. 5. Operation count results (scaled by the local matrix evaluation operation count) of the different evaluation strategies for a structured quadrilateral mesh. The boxes with encircled tags S (sum-factorisation), L (local matrix) or G (global matrix) indicate the optimal strategy for the corresponding range of *P*.

tal boundary modes will decrease. As a results, when neglecting the cost of assembly of the local matrix approach, the complexity of the global matrix evaluation will asymptotically approach to the complexity of the local matrix evaluation for $P \to \infty$. This can also be observed in Fig. 5.

3.2.2.2. Triangular expansions. For unstructured triangular meshes, it is more difficult to estimate the non-zero entries of the global matrix $\boldsymbol{M}$ as it depends on the distribution of the triangles within the mesh. Estimates have been made in [14] and the results are presented in Table 2 and depicted in Fig. 6. A similar trend as for structured quadrilateral meshes can be observed, but the advantage of the global matrix approach is even greater in the triangular case. This can be explained by the bigger boundary modes to interior modes ratio for triangles, resulting in greater dominance of the multiplicity of global DOFs.

Remark 3. Although the Helmholtz and mass matrix operator share an identical operation count for both the local and global matrix approach, their results differ from the backward transformation and inner product operator. This is because one dimension of the backward transformation and inner product is in terms of the quadrature points, which are not being assembled. The global matrix evaluation then only benefits from assembly in one direction. However, it appears that this assembly does not lead to a reduction in non-zero entries such that the local matrix approach and global matrix operation have similar operation count for these operators. On the other hand, the assembly does decrease the rank of the operator.

3.2.3. Optimal strategy

Based upon the operation count estimates summarised in Figs. 5 and 6, one may conclude that for low-order expansions the global matrix evaluation strategy is superior, while for high-order expansions, the sum-factorisation technique is preferred.

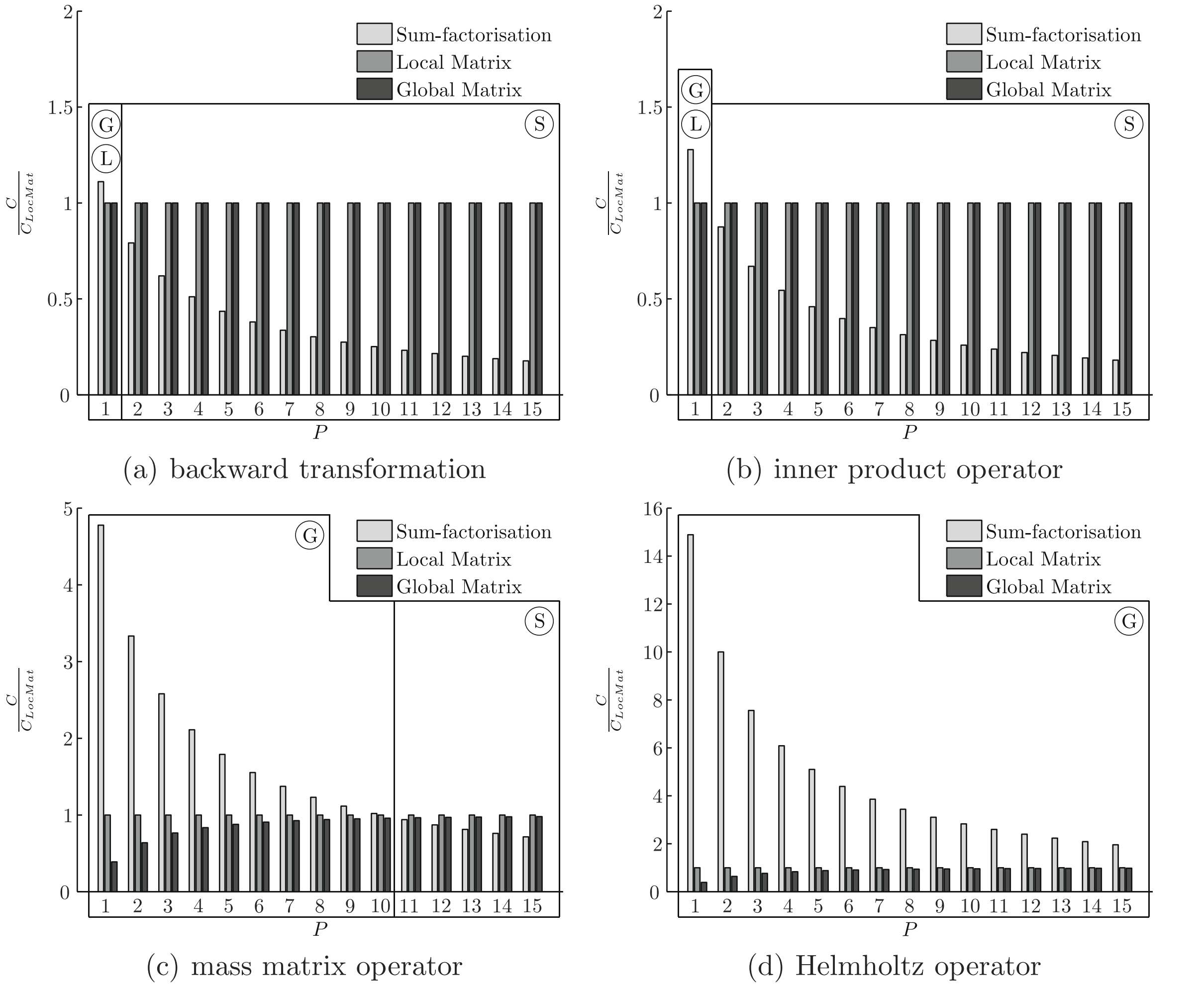


Fig. 6. Operation count results (scaled by the local matrix evaluation operation count) of the different evaluation strategies for an unstructured triangular mesh. The boxes with encircled tags S (sum-factorisation), L (local matrix) or G (global matrix) indicate the optimal strategy for the corresponding range of *P*.

Table 2
Operation count estimates of the local matrix and global matrix approach to evaluate a global bi-linear form on a structured quadrilateral mesh ($|\mathcal{E}^{1d}| \times |\mathcal{E}^{1d}| = |\mathcal{E}|$ elements) and an unstructured triangular mesh ($|\mathcal{E}|$ elements).

		Local matrix	Global matrix	$\lim_{|\mathcal{E}|\to\infty} \frac{\text{GlobMat}}{\text{LocMat}}$
Quad	DOFs	$|\mathcal{E}|(P+1)^2$	$(|\mathcal{E}^{1d}|P+1)^2$	$\left(1-\frac{1}{P+1}\right)^2$
	operation count	$2|\mathcal{E}|(P+1)^4$	$2(|\mathcal{E}^{1d}|P(P+2)+1)^2$	$\left(1-\frac{1}{(P+1)^2}\right)^2$
Tri	DOFs	$\frac{1}{2}|\mathcal{E}|(P+1)(P+2)$	$\frac{1}{2}|\mathcal{E}|P^2$	$\frac{P^2}{(P+1)(P+2)}$
	Operation count	$2|\mathcal{E}|\left(\frac{1}{2}(P+1)(P+2)\right)^2$	$2|\mathcal{E}|\frac{1}{4}(P^4+6P^3+7P^2)$	$\frac{P^4+6P^3+7P^2}{((P+1)(P+2))^2}$

3.3. Computational cost based on run-time

The previous section indicates that the application of different evaluation strategies depending on *P* leads to a reduction in operation count. Here, we analyse whether this reduction in operation count leads to more efficient algorithms by directly comparing the resulting run-time. This may not be guaranteed as the efficiency of a certain implementation is not only quantified by the number of floating point operations. Various other factors such as the number of memory references, memory access time, caching effects, data structures and chip architecture do play an important role as well. Furthermore, the role of

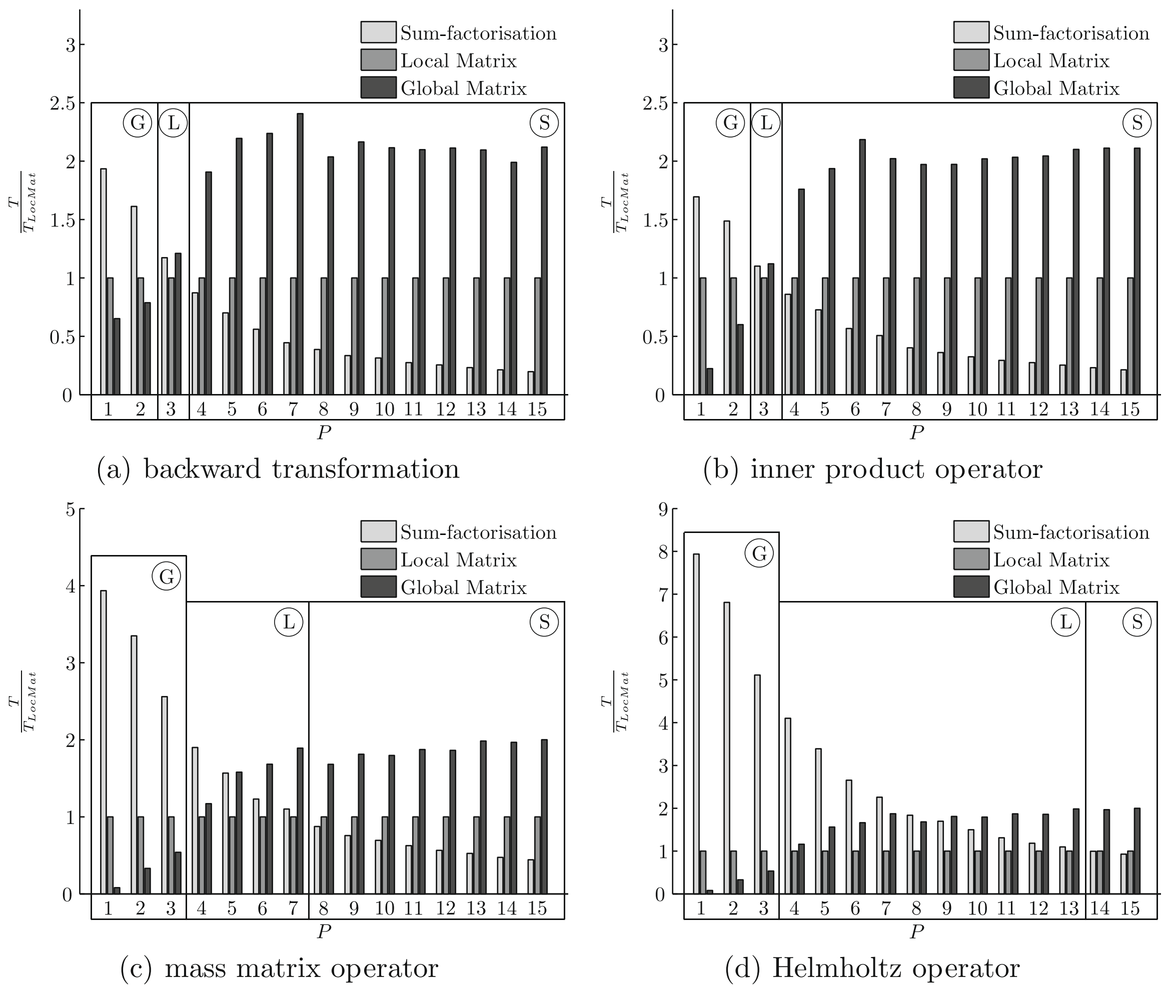


Fig. 7. Computational cost (*i.e.* run-time scaled by the local matrix evaluation run-time) of the different evaluation strategies for a structured quadrilateral mesh of 1000 elements. The boxes with encircled tags S (sum-factorisation), L (local matrix) or G (global matrix) indicate the optimal strategy for the corresponding range of P.

possibly optimised linear algebra packages such as BLAS[2] should not be forgotten. As it is a cumbersome, if not impossible, task to estimate the cost of these separate parameters, the efficiency of the different strategies will be assessed by comparing the total computational cost (quantified by the actual run-time) of the different strategies.

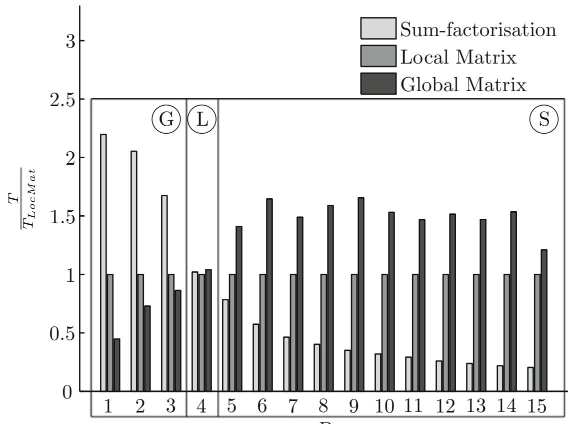
The results are summarised in Fig. 7 and 8. We will now compare these results with Figs. 5 and 6, respectively, leading to the following two important observations:

- The performance benefit of the sum-factorisation technique is reduced, shifting the break-even point between the elemental approaches to the right in favour of the local matrix approach. This can probably be explained by the fact that the sum-factorisation technique requires temporary storage, which induces some additional cost from a computational point of view.
- Although the global matrix technique is still preferable for low-orders, it rapidly becomes excessively expensive for higher orders. This is most likely due to the inability to exploit the locality of the data which cancels the reduction in operation count.

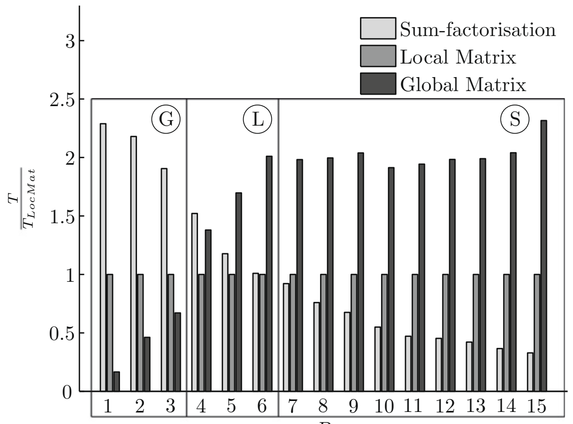
Both these factors contribute to the rise of an intermediate regime between low- and high-order expansions, where the local matrix approach now is the optimal strategy. This is clearly visible in both Figs. 7 and 8.

All tests presented in this section were run on an Intel MacBook Pro (2.33 GHz dual core processor, 2 GB RAM) and the performance tests were based upon the implementation of the different strategies within the Nektar++ framework.[3] The

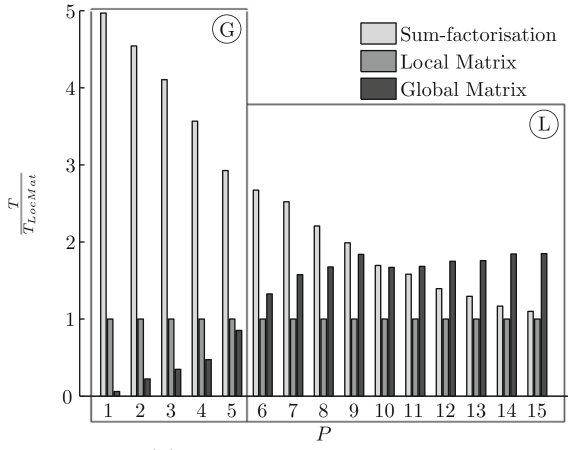
[2] http://www.netlib.org/blas.
[3] http://www.nektar.info.



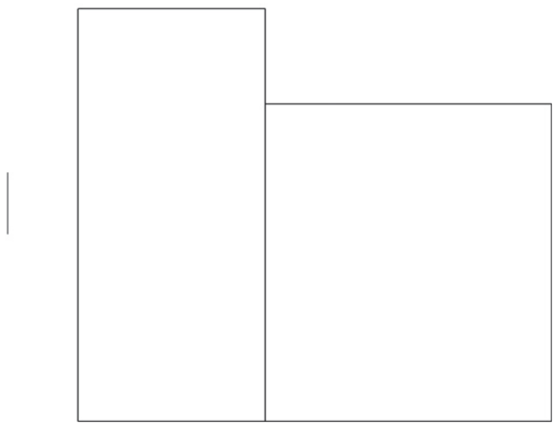
(a) backward transformation



(b) inner product operator



(c) m



computational kernel for the sum-factorisation technique and local matrix approach was based upon the reference implementation of the BLAS routines `dgemm` and `dgemv`, respectively.[4] The global matrix evaluation was implemented using the `dcsrmm` routine of the NIST sparse BLAS library.[5] Validation tests were run on machines with different specifications, and although the results may differ slightly in terms of the break-even points, the general trends and principles observed have been confirmed.

3.4. Lessons learned

In order to efficiently implement the spectral/*hp* element method for a broad range of polynomial orders, one should distinguish three different regimes:

- a low-order regime where the global matrix approach is most efficient,
- an intermediate order regime where the local matrix approach is most efficient, and
- a high-order regime where the sum-factorisation approach is most efficient.

Note that efficiency is defined here in a computational sense, *i.e.* minimising the actual run-time. Remember that we have assumed the use of tensorial expansion bases in the analysis. For non-tensorial expansions such as the nodal triangular spectral/*hp* expansions [9], only the first two regimes will be applicable as the sum-factorisation technique cannot be applied.

Consequently, non-tensorial expansion will not benefit from the observed performance gain due to the sum-factorisation technique in case of high-order expansions.

It can be observed that selecting a non-optimal evaluation strategy can lead to very inefficient code. When for example evaluating the Helmholtz operator for a fourth-order triangular spectral/*hp* expansion, applying the sum-factorisation technique rather than the global matrix approach will increase the run-time by a factor 15. For a linear expansion, this factor has even been observed to be as high as 150.

The break-even points between the different regimes will in general depend on:

- the operator to be evaluated,
- the shape of the element (quadrilateral versus triangle), and
- the computer on which the code is run.

Although the theoretical operation count may have given an indication of these regimes, the results in Section 3.3 show the importance of performance test to determine the computer-specific break-even points, especially when operating in the intermediate regime between low and high-order. For example, based upon operation count, Fig. 5(c) would suggest that sum-factorisation is the optimal technique to evaluate the mass matrix for a fifth-order expansion. However, Fig. 7(c) shows that the application of the local matrix approach leads to a reduction in run-time with a factor 1/3. This also supports the idea of a self-tuning library (such as the ATLAS[6] implementation of BLAS): in order to obtain optimal performance, a set of tests should be run during installation in order to determine the machine-specific break-even points.

4. Optimal spectral/*hp* discretisations

Now we have determined the optimal implementation for both low and high-order expansions, we may ask the following question: given the most efficient implementation, what is the optimal spectral/*hp* discretisation for a given error tolerance? We define the optimal discretisation as the (h,P)-pair – the specific combination of mesh size h and polynomial order P – which requires the minimal run-time to approximate the exact solution up to a predefined accuracy. Answers to this question have been presented before in [18–22]. In these previous works, the computational cost of the algorithms have been estimated by analytical models. However, based upon the results of the previous section, we believe that an analysis from a purely analytic point of view may not be able to correctly model all factors that make up the actual run-time. Therefore, we adopt a fully computational approach where we base our analysis on the measured run-time of a set of performance test. In addition, note that we do not aim to provide a universal statement to answer this question as it will highly depend of the problem under consideration. Instead we have identified a few examples and demonstrate how the results of the previous section may influence the discussion. Therefore, we have chosen to solve the scalar Helmholtz equation for four different test cases.

4.1. Test problem: the scalar Helmholtz equation

The two-dimensional scalar Helmholtz problem on a domain Ω is given by the equation

$$\nabla^2 u(x,y) - \lambda u(x,y) = f(x,y), \quad \lambda \geqslant 0. \tag{39}$$

The problem is supplied with Dirichlet boundary conditions on the entire boundary of the domain, *i.e.* $u(x,y)|_{\partial\Omega} = g_{\mathcal{D}}(x,y)$ and we for simplicity we assume $\lambda = 1$. We have chosen not to consider a more complex test case as we believe the simplicity of this problems enables us to unambiguously investigate the influence of the mesh-size h and polynomial order P. If for example selecting a time-dependent problem such as the advection–diffusion equation, the time-step Δt will depend on the mesh-size h through the CFL-condition. It can be appreciated that this additional dependency will quickly complicate the analysis. Therefore, it should be kept in mind that the presented results cannot simply be extrapolated to time-dependent problems as this will require a careful analysis.

To solve Eq. (39) we follow a standard Galerkin procedure to obtain the weak form: Find $u \in \mathcal{U}$ such that

$$\int_\Omega \nabla v \cdot \nabla u \, d\boldsymbol{x} + \int_\Omega vu \, d\boldsymbol{x} = -\int_\Omega vf \, d\boldsymbol{x}, \quad \forall v \in \mathcal{V}, \tag{40}$$

where $\mathcal{U}$ and $\mathcal{V}$ are a suitably chosen trial and test space, respectively. In order to impose the Dirichlet boundary conditions we adopt a lifting strategy where the solution is decomposed into a known function, u^D and an unknown homogeneous function u^H, *i.e.*

$$u(\boldsymbol{x}) = u^H(\boldsymbol{x}) + u^D(\boldsymbol{x}). \tag{41}$$

Here u^D is satisfying the Dirichlet boundary conditions, $u^D(\partial\Omega_D) = g_D$ while u^H is homogeneous on the Dirichlet boundary, $u^H(\partial\Omega_D) = 0$. Subsequently, as part of the discretisation process, the solution is expanded in terms of a globally C^0-continuous expansion basis as

[6] http://math-atlas.sourceforge.net.

$$u(\boldsymbol{x}) = \sum_{i\in\mathcal{N}} \Phi_i(\boldsymbol{x})\hat{u}_i^g = \sum_{i\in\mathcal{N}^{\mathcal{H}}} \Phi_i^{\mathcal{H}}(\boldsymbol{x})\hat{u}_i^{\mathcal{H}} + \sum_{i\in\mathcal{N}} \Phi_i(\boldsymbol{x})\hat{u}_i^{\mathcal{D}}, \tag{42}$$

where the set $\{\Phi^H\}$ consists of the basis-functions having no support on the Dirichlet boundary. Note that $\mathcal{N}$ now refers to the index set of the global degrees of freedom. Finally, employing the same expansion basis $\{\Phi^H\}$ to span the test space $\mathcal{V}$ results in the following discrete global system to be solved

$$\sum_{j\in\mathcal{N}^{\mathcal{H}}} h(\Phi_i^{\mathcal{H}}, \Phi_j^{\mathcal{H}})\hat{u}_j^{\mathcal{H}} = -(\Phi_i^{\mathcal{H}}, f) - \sum_{j\in\mathcal{N}} h(\Phi_i^{\mathcal{H}}, \Phi_j)\hat{u}_j^{\mathcal{D}}, \quad \forall i \in \mathcal{N}^{\mathcal{H}}, \tag{43}$$

where the bi-linear form $h(u,v) = \int \nabla u \cdot \nabla v d\boldsymbol{x} + \int uv d\boldsymbol{x}$ corresponds to the Helmholtz operator and (u, v) is defined as the inner product $(u, v) = \int uv d\boldsymbol{x}$.

From an implementational point of view, Eq. (43) can typically be solved in four major steps (thereby neglecting the steps involving linear combinations of vectors):

(1) Calculate the inner product of the forcing function $(\Phi_i^{\mathcal{H}}, f),\ \forall i \in \mathcal{N}^{\mathcal{H}}$.
(2) Calculate the Dirichlet forcing $\sum_{j\in\mathcal{N}} h(\Phi_i^{\mathcal{H}}, \Phi_j)\hat{u}_j^{\mathcal{D}},\ \forall i \in \mathcal{N}^{\mathcal{H}}$.
(3) Solve the linear system of type $\boldsymbol{H}\hat{\boldsymbol{u}}^{\mathcal{H}} = \boldsymbol{f}$ where $\boldsymbol{H}$ is the global Helmholtz matrix.
(4) Transform the coefficients back to physical space, *i.e.* $u(\boldsymbol{x}_i) = \sum_{j\in\mathcal{N}} \Phi_j(\boldsymbol{x}_i)\hat{u}_j^g,\ \forall i \in \mathcal{Q}$.

We have adopted an implementation strategy where the first two operations are evaluated with respect to the entire set of global basis functions, *i.e.* $(\Phi_i, f),\ \forall i \in \mathcal{N}$ rather than $(\Phi_i^{\mathcal{H}}, f),\ \forall i \in \mathcal{N}^{\mathcal{H}}$. Because of this, it can be appreciated that the evaluation of steps (1) and (2), respectively, correspond to the inner product and Helmholtz operator as described in Section 3. In addition, it can be recognised that step (4) is the operation that we have defined as a backward transformation. As a results, these three sub-steps of the solution algorithm may benefit from the optimisation strategies introduced in Section 3. It is therefore through these routines that the influence of the different implementation strategies on the definition of optimal *hp*-discretisations may become apparent. Finally, note that when adopting an iterative solution method such as the conjugate gradient method, the implementation of the third step can also be based on the different implementation strategies. However, due to the limited size of the global system in two-dimensional, we have adopted a direct solution method based upon the static condensation of the interior degrees of freedom [7]. The resulting Schur complement is reordered for minimal bandwidth and solved using the LAPACK routines `dptrf` and `dptrs`.[7]

4.2. Test problem 1: quadrilateral spectral/hp discretisations for a smooth solution

The first example we consider is a smooth solution on the unit square $\Omega = [0,1] \times [0,1]$. The forcing function f and the Dirichlet boundary conditions $g_{\mathcal{D}}$ are chosen such that the exact solution satisfies

$$u_{ex}(x,y) = \sin(10\pi x)\cos(10\pi y). \tag{44}$$

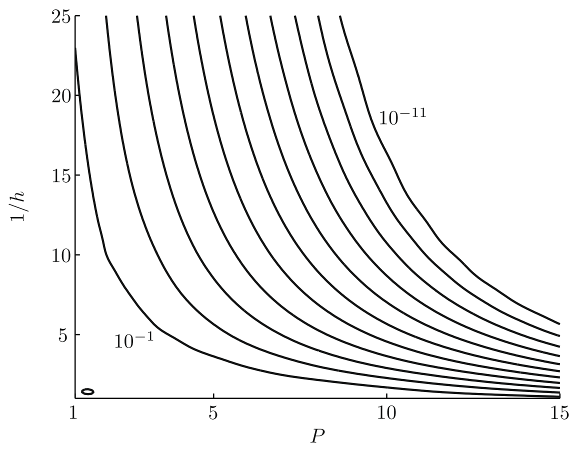
Our aim is to find the quadrilateral *hp*-discretisation which minimises the run-time for a certain error tolerance. Therefore, we let the mesh-size between and the polynomial order, respectively, vary between $1 \leqslant 1/h = |\mathcal{E}^{1d}| \leqslant 25$, $1 \leqslant P \leqslant 15$ and solve the corresponding Helmholtz equation for all 375 possible (h,P) combinations. Note that the spatial uniformity of the solution justifies the uniform adaptation of the expansion. The L_2 approximation error defined as

$$\|\epsilon\|_{L_2} = \left[\int_{\Omega} (u_{ex} - u)^2 d\boldsymbol{x}\right]^{\frac{1}{2}}, \tag{45}$$

is depicted in Fig. 9(a) for all discretisations. Predictably, low-order expansions on a coarse mesh exhibit a large error while high-order expansions on a fine mesh lead to the most accurate results. The computational cost – quantified by the run-time – of every (h,P) pair is plotted in a similar style in Fig. 9(b) where the error plots are overlaid. We here show the run-time result based upon the optimal implementation, *i.e.* an implementation where depending on the polynomial order P we have selected the most efficient evaluation strategy for each individual operator following the results of Section 3.3. Although both the error and cost contours follow a similar trend, they are not parallel. Note that although all data are essentially discrete, we have interpolated the data in the presentation of these results to obtain a continuous representation.

If we now fix the error tolerance to 10%, we can ask how these plots may help us to find the optimal discretisation (h,P) satisfying this tolerance? Therefore, we want to know which point on the 10% error contour line depicted in Fig. 9(a) induces the minimal run-time. Therefore, we can use the arc-length of this contour line as the horizontal axis and plot the corresponding computational cost from Fig. 9(b) as a function of this arc-length, *i.e.* we extract the data along the solid black line in Fig. 9(b). This is shown in Fig. 10(a). Besides the cost due to the optimal implementation, this figure includes different lines all corresponding to a different interpretation of the computational cost. The dashed line defines the computational cost as the total number of global degrees of freedom while the other lines are due to a single implementation strategy using either a

[7] http://www.netlib.org/lapack.



(a) Error contours $||\epsilon||_{L_2} = 10^i\,(1 \leq i \leq 11)$

global matrices, local matrices or the sum-factorisation approach. Note that all four lines lead to a different minimum and that the one due to the optimal implementation truly minimises the run-time of all the different implementation strategies. Table 3 (Test problem 1) summarises the *hp*-discretisations to which these minima relate back. It appears that *hp*-expansion employing 3×3 elements and a sixth-order expansion can be regarded as the optimal discretisation for our computational test. Note that this expansion is notably different from the one that minimises the number of degrees of freedom for the given error tolerance. For the example under consideration, this would suggest a spectral-type approach employing only a single element but with a sufficiently high-order $P > 15$. As expected from Section 3, an implementation based upon global matrices has a tendency towards *h*-type low-order finite elements while the use of sum-factorisation based routines would advocate a more spectral approach.

Fig. 10(a) also illustrates the importance of supporting different implementation strategies within a single code. As the curve due to the optimal implementation strategy could be considered as the envelope of the three single-implementation curves, selecting another *hp*-discretisation along the horizontal axis does not drastically increases the computational cost. However, adopting only a single implementation strategy it can be seen that the optimum is much sharper and deviating from this optimal discretisation is severely penalised.

In Fig. 11, we again have plotted the run-time due to the optimal implementation strategy along the 10% error contour. In addition, we have indicated which fraction of the run-time is due to the solution of the linear system, *i.e.* the third step in the solution process of the Helmholtz equations as explained in Section 4.1. It can be observed that along this error contour, solving the linear system is more efficient for the expansions that combine fewer elements with higher *P*. This may be a direct consequence of the smaller number of global degrees of freedom as observed in Fig. 10(b). As a result, it will be the three remaining steps in the evaluation process which shift the minimum towards a lower order expansion employing a more refined mesh. When using an iterative rather than direct solver, a similar shifting towards lower order can be expected as iterative solvers essentially encompass these three steps.

We can follow a similar analysis for a broad range of error tolerances. This leads to the *path of minimal run-time* as depicted in Fig. 10. This figure does confirm the well-known feature of exponential *p*-convergence of the spectral/*hp* element method for smooth solutions. When aiming for a high-accurate approximation, one should increase the polynomial order rather than refining the mesh. It appears that for this example, a 3×3 mesh is the optimal *h*-discretisation and the polynomial order *P* should be varied according to the desired accuracy. This typical behaviour has been acknowledged before and it

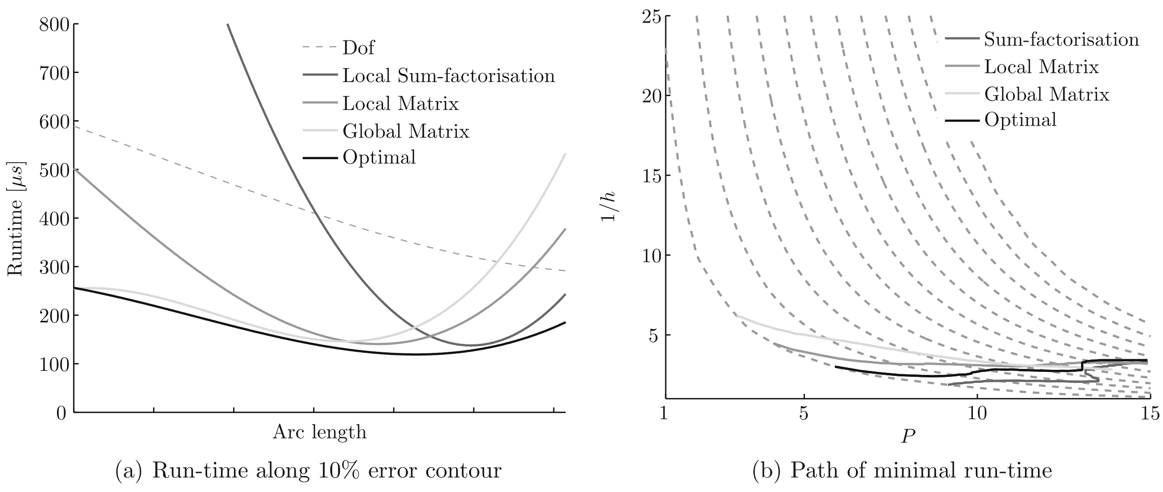


Fig. 10. Minimal run-time at fixed error-level of the different quadrilateral spectral/*hp* discretisations used for approximating the Helmholtz problem with smooth exact solution (44).

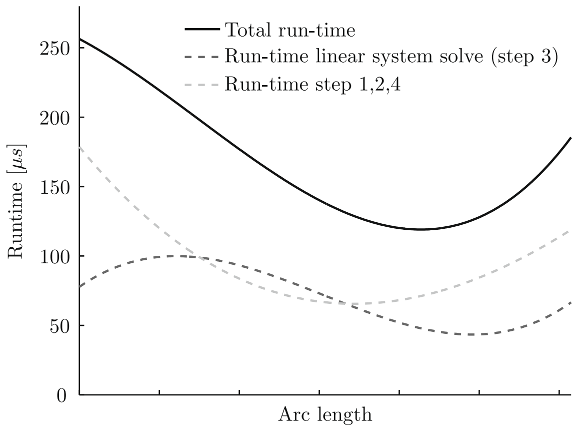


Fig. 11. Decomposition of the run-time due to the optimal (hybrid) implementation strategy along the 10% error contour when approximating the Helmholtz problem with smooth exact solution (44).

is widely appreciated that the spectral/*hp* element method is particularly efficient for obtaining high accuracy in smooth problems [8,7]. However, by focusing on a engineering accuracy of 10%, we have shown that the high-order methods can also be justified for relatively low accurate approximations.

Finally, we would like to couple this path of minimal run-time to an earlier observation made by Gottlieb and Orszag. In [10], they showed that for a one-dimensional problem with exact solution

$$u(x) = \sin(M\pi x) \quad \text{on } [-1,1], \tag{46}$$

a single-element spectral expansion should retain at least $N > M\pi$ modes in order to achieve exponential *p*-convergence. Translated to the two-dimensional problem under consideration, this would suggest that for exponential *p*-convergence, the multi-elemental spectral/*hp* element discretisation should satisfy

$$N > 10\pi\frac{h}{2}, \tag{47}$$

where $N = P + 1$. This condition is graphically represented in Fig. 12 as the area above the "Gottlieb–Orszag threshold". Heuristically, this may lead to the following discretisation strategy: use *h*-type refinement until crossing this "Gottlieb–Orszag threshold" and subsequently increase the polynomial order *P* according to the desired accuracy. This leads to the convergence path also shown in Fig. 12. Although this approach may provide a simple rule of thumb, the results show that the resulting convergence path is different from the path of minimal run-time defined earlier and hence will be computationally less efficient. The problem is that in this approach, it has been assumed that one should follow an *h*-type refinement strategy (with $P = 1$) in the unresolved regime. Although this indeed leads to the least expensive point on the Gottlieb–Orszag

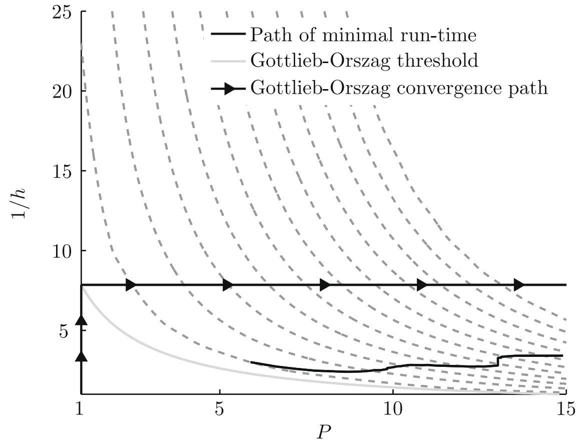


Fig. 12. Possible path of convergence when following the argument presented by Gottlieb and Orszag when approximating the Helmholtz problem with smooth exact solution (44).

threshold in Fig. 12, this point certainly does not minimise the run-time on the corresponding error contour. Instead, our analysis shows that for minimal run-time one should combine both ideas of *h*-refinement and *p*-enrichment to select an initial discretisation along the Gottlieb–Orszag threshold. Hereafter, the polynomial order *P* can be increased for higher accuracy.

4.3. Test problem 2: quadrilateral spectral/hp discretisations for a smooth solution with high wave-number

As a second example, we consider the Helmholtz equation with a similar solution but with a higher wave-number

$$u(x,y) = \sin(20\pi x)\cos(20\pi y). \tag{48}$$

Obviously, the computational cost for each specific spectral/*hp* expansion is the same as in the previous example but it now will require more degrees of freedom to obtain a specific error tolerance. Following a similar approach as in the previous example, it can be derived that for an engineering accuracy of 10%, the optimal *hp*-discretisation minimising the run-time again comprises a sixth-order expansion but now on a 6×6 mesh, see also Table 3. As in the previous case, the most efficient way of enhancing the accuracy is by increasing the polynomial order and keeping the mesh fixed. The analogy with the previous example seems to suggest that for the type of exact solution under consideration, *i.e.*

$$u(x,y) = \sin(M\pi x)\cos(M\pi y), \tag{49}$$

the optimal quadrilateral *h*-discretisation consists of $\frac{3M}{10} \times \frac{3M}{10}$ elements, independent of the desired accuracy. Hence, it is the polynomial order that should be varied accordingly to satisfy a predefined error tolerance.

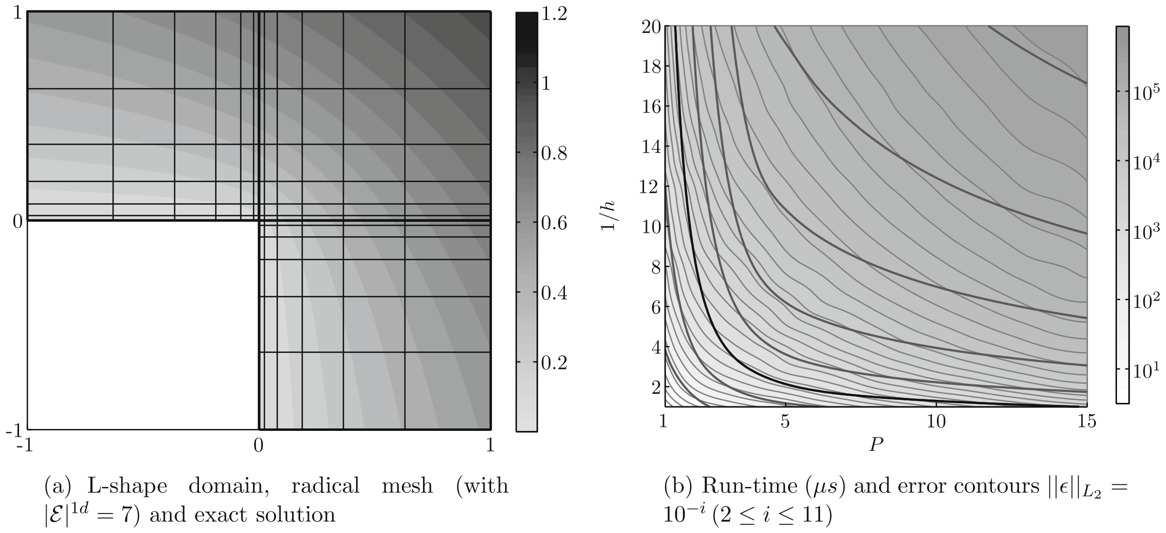
4.4. Test problem 3: triangular spectral/hp discretisations for a smooth solution

Next we consider exactly the same problem as in Section 4.2 for the first test problem. However, rather then using structured quadrilateral meshes we now use unstructured triangular meshes. Using the same analysis as for the quadrilaterals examples, we have observed that for a 10% error, again a sixth-order spectral/*hp* expansion with characteristic mesh-size $h = 1/5$ – which corresponds to *38* triangles – is the computationally most efficient *hp*-discretisation (see also Table 3). The run-time associated to this optimal triangular *hp*-discretisation appears to more than twice the run-time needed for the optimal quadrilateral expansion. As a result, for the uniform test-case under consideration, the quadrilateral sixth-order expansion employing 3×3 elements can be considered as the true optimal spectral/*hp* expansion. However, we would like to remark that this apparent superiority of the quadrilateral expansion is reinforced by the tensorial structure of the exact solution. Indeed, if we rotate the exact solution with 30°, the overhead of the triangular expansion is reduced by 20%. Regarding the convergence of the error in this triangular case, we can draw analogous conclusions as for the quadrilateral case.

4.5. Test problem 4: quadrilateral spectral/hp discretisations for an irregular solution

Finally, we include an example which exact solution is not infinitely smooth. Therefore, we consider the L-shape domain problem shown in Fig. 13. The forcing function *f* and Dirichlet boundary conditions are chosen such that the Helmholtz Eq. (39) yields the exact solution

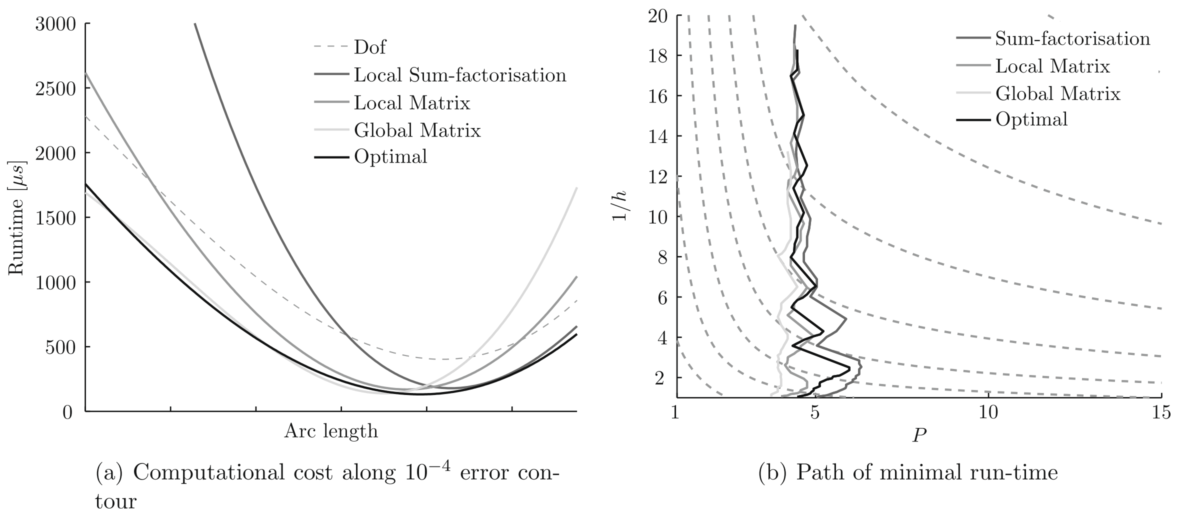
$$u(x(r,\theta),y(r,\theta)) = r^{\frac{2}{3}}\sin\left(\frac{2}{3}\theta + \frac{\pi}{3}\right), \tag{50}$$



(a) L-shape domain, radical mesh (with $|\mathcal{E}|^{1d} = 7$) and exact solution

(b) Run-time (μs) and error contours $||\epsilon||_{L_2} = 10^{-i}$ $(2 \le i \le 11)$

Fig. 13. Quadrilateral spectral/*hp* discretisations to approximate the Helmholtz problem with exact solution (50).



(a) Computational cost along 10^{-4} error contour

(b) Path of minimal run-time

Fig. 14. Quadrilateral spectral/*hp* discretisations to approximate the Helmholtz problem with exact solution (50).

where (r,θ) are the traditional polar coordinates. Because of the term $r^{\frac{2}{3}}$, the gradient of the solution will exhibit a singularity at the re-entrant corner. This problem has been extensively studied in a FEM and *hp*-FEM context, often with an emphasis on adaptive refinement, see *e.g.* [23]. In this study, we do not consider adaptivity but adopt a discretisation strategy where we account for the locality of the singularity by using a radical rather than equispaced mesh. The grid-points in each interval [0,±1] of a radical mesh are located at

$$x_i = \pm\left(\frac{i}{|\mathcal{E}|^{1d}}\right)^{\beta}, \quad i = 0, 1, \ldots, |\mathcal{E}|^{1d}, \tag{51}$$

where the parameter β is chosen as $\beta = 3$ (according to [24]) and $|\mathcal{E}|^{1d}$ is the number of elements in one dimension. The error and cost contours are depicted in Fig. 13(b). Note that the error contours follow a different pattern than for the cases with a smooth solution. This problem therefor has an entirely different path of minimal run-time, see Fig. 14(b). There are various remarkable features in this figure. First it can be seen that for a given error tolerance, all strategies appear to suggest almost the same (h,P) discretisation. This includes the discretisation which minimises the number of degrees-of-freedom rather than the run-time. This may be appreciated by considering Fig. 14(a) which shows that the minimum – based upon DOFs – along the 10^{-4} error contour is now much sharper than for the smooth test-problems, see for example Fig. 10(a). From Fig. 14(b) it appears that the effect of the different implementation strategies do not cause significantly different run-time to overcome the overhead of additional DOFs. As a result, all minima seem to coincide around the same discretisation. Secondly, it can be observed that the path of minimal run-time converges along the *h*-direction, rather than the *P*-direction. This is plausible as the spectral/*hp* element method does not exhibit exponential error-convergence with respect to the polynomial order because of the singularity. In addition the use of a radical mesh, which has been shown to be optimal for *h*-type FEM in this context, and so may contribute to this observation. Finally, Fig. 14(b) clearly indicates that a fifth-order expansion is now optimal and that the mesh-size should be varied according to the desired accuracy. Although this type of *h*-type convergence for non-smooth problems is typically associated to low-order finite element methods ($P = 1$, or sometimes $1 \leqslant P \leqslant 3$), the resulting fifth-order optimum indicates that high-order finite element methods also have their use in solving singular problems.

5. Summary

We have shown that in order to implement the spectral/*hp* element method for the broad range of polynomial orders ($1 \leqslant P \leqslant 15$), a spectral/*hp* element code should ideally support three different implementation strategies to evaluate the finite element operators. This allows a hybrid strategy based upon the polynomial order. For low-order expansions, as is common practice we have shown that the evaluation using global matrices is most efficient while for high order expansions, one should preferably employ the sum-factorisation technique. If operating in the intermediate regime between low- and high-order, the evaluation using local matrices is often the most efficient option. Furthermore, we demonstrated that the break-even points between these different polynomial regimes depend on the operator to be evaluated, the shape of the element and the computer on which the code is run. We have presented both theoretical estimates as well as computational test confirming this behaviour for different two-dimensional finite element operators.

In the second part of the paper, we have investigated how to select the parameters (h,P) of a spectral/*hp* discretisations in order to minimise the run-time when solving an elliptic problem up to predefined accuracy. As expected, the numerical results indicate that in case of smooth solutions, one should fix the mesh and vary the polynomial order according to the desired accuracy (*p*-convergence). In addition, our computational investigation has however highlighted the not quite so intuitive result that for a low error level of 10% a reasonably coarse mesh with a sixth-order spectral/*hp* expansions minimised the run-time. For non-smooth solutions on the other hand, and consistent with theory, we observed that the run-time can be minimised by fixing the polynomial order of the expansion and refining the mesh according to the desired error tolerance (*h*-convergence). However, for the non-smooth test problem under consideration, we observed that a polynomial order of as high as $P = 5$ was optimal, thereby promoting the use of high-order expansions for problems with corner-type singularities, at least when using a radical mesh distribution.

Acknowledgments

The authors would like to acknowledge the insightful input of Professor Paul Kelly of Imperial College London. S.J.S. and P.E.J.V. would like to acknowledge the support under an EPSRC Advance Research Fellowship and R.M.K. would like to acknowledge support from the Leverhulme Trust.

References

[1] S.J. Sherwin, G.E. Karniadakis, Tetrahedral *hp* finite elements: algorithms and flow simulations, J. Comput. Phys. 124 (1996) 14–45.
[2] J. Hesthaven, T.C.E. Warburton, Nodal high-order methods on unstructured grids: I. Time-domain solution of maxwell's equations, J. Comput. Phys. 181 (2002) 186–221.
[3] P.-E. Bernard, J.-F. Remacle, R. Combien, V. Legat, K. Hillewaert, High-order discontinuous Galerkin schemes on general 2D manifolds applied to the shallow water equations, J. Comput. Phys. 228 (2009) 6514–6535.
[4] O.C. Zienkiewicz, R.L. Taylor, The Finite Element Method, fourth ed., McGraw-Hill, New York, 1989.
[5] T.J.R. Hughes, The Finite Element Method, Prentice-Hall, New Jersey, 1987.
[6] B. Szabó, I. Babuška, Finite Element Analysis, Wiley, New York, 1991.
[7] G.E. Karniadakis, S.J. Sherwin, Spectral/*hp* Element Methods for Computational Fluid Dynamics. Numerical Mathematics and Scientific Computation, second ed., Oxford University Press, Oxford, 2005.
[8] M.O. Deville, P.F. Fischer, E.H. Mund, High-order Methods for Incompressible Fluid Flow. Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, Cambridge, 2002.
[9] J.S. Hesthaven, T. Warburton, Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications, Springer Texts in Applied Mathematics, vol. 54, Springer-Verlag, New York, 2008.
[10] D. Gottlieb, S.A. Orszag, Numerical Analysis of Spectral Methods: Theory and Applications, CBMS-NSF, Society for Industrial and Applied Mathematics, Philadelphia, 1977.
[11] J.F. Remacle, J.E. Flaherty, M.S. Shepard, An adaptive discontinuous Galerkin technique with an orthogonal basis applied to compressible flow problems, SIAM Rev. 45 (2003) 55–73.
[12] T.J.R. Hughes, J.A. Cottrell, Y. Bazilevs, Isogeometric analysis: cad, finite elements, nurbs, exact geometry and mesh refinement, Comput. Meth. Appl. Mech. Eng. 194 (39-41) (2005) 4135–4195.
[13] Moshe Dubiner, Spectral methods on triangles and other domains, J. Sci. Comput. 6 (4) (1991) 345–390.
[14] Babak Bagheri, L. Ridgway Scott, Shangyou Zhang, Implementing and using high-order finite element methods, Finite Elem. Anal. Des. 16 (3–4) (1994) 175–189. 186260.
[15] R.C. Kirby, M. Knepley, L.R. Scott, Evaluation of the action of finite element operators, Technical Report TR-2004-07, University of Chicago, 2004.
[16] P.E.J. Vos, S.J. Sherwin, R.M. Kirby, From *h* to *p* efficiently, Technical Report UUCS-09-007, University of Utah, 2009.
[17] S.A. Orszag, Spectral methods for problems in complex geometries, J. Comput. Phys. 37 (1) (1980) 70–92.
[18] E.M. Rønquist, Optimal spectral element methods for unsteady three-dimensional incompressible Navier–Stokes equations, Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, 1988.

[19] P. Fischer, D. Gottlieb, On the optimal number of subdomains for hyperbolic problems on parallel computers, Int. J. Supercomputer Appl. High Perform. Comput. 11 (1996) 65–76.
[20] J.S. Hesthaven, A stable penalty method for the compressible Navier–Stokes equations: Ii. One-dimensional domain decomposition schemes, SIAM J. Sci. Comput. 18 (3) (1997) 658–685.
[21] J.S. Hesthaven, A stable penalty method for the compressible Navier–Stokes equations: Iii. Multidimensional domain decomposition schemes, SIAM J. Sci. Comput. 20 (1) (1998) 62–93.
[22] C.E. Wasberg, D. Gottlieb, Optimal decomposition of the domain in spectral methods for wave-like phenomena, SIAM J. Sci. Comput. 22 (2) (2000) 617–632.
[23] I. Babuška, M. Suri, The *p* and *h*–*p* versions of the finite element method, basic principles and properties, SIAM Rev. 36 (4) (1994) 578–632.
[24] P. Seshaiyer, M. Suri, *hp* submeshing via non-conforming finite element methods, Comput. Meth. Appl. Mech. Eng. 189 (3) (2000) 1011–1030.