Journal of Computational Physics 346 (2017) 437–448

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Factorizing the factorization – a spectral-element solver for elliptic equations with linear operation count

Immo Huismann [a,b,*], Jörg Stiller [a,b], Jochen Fröhlich [a,b]

[a] Institute of Fluid Mechanics, TU Dresden, George-Bähr-Straße 3c, 01062 Dresden, Germany
[b] Center for Advancing Electronics Dresden (cfaed), Helmholtz-Straße 10, 01062 Dresden, Germany

ARTICLE INFO

Article history:
Received 29 January 2016
Received in revised form 2 June 2017
Accepted 6 June 2017
Available online 13 June 2017

Keywords:
Spectral-element method
Elliptic equations
Substructuring
Static condensation

ABSTRACT

The paper proposes a novel factorization technique for static condensation of a spectral-element discretization matrix that yields a linear operation count of just $13N$ multiplications for the residual evaluation, where N is the total number of unknowns. In comparison to previous work it saves a factor larger than 3 and outpaces unfactored variants for all polynomial degrees. Using the new technique as a building block for a preconditioned conjugate gradient method yields linear scaling of the runtime with N which is demonstrated for polynomial degrees from 2 to 32. This makes the spectral-element method cost effective even for low polynomial degrees. Moreover, the dependence of the iterative solution on the element aspect ratio is addressed, showing only a slight increase in the number of iterations for aspect ratios up to 128. Hence, the solver is very robust for practical applications.

© 2017 Elsevier Inc. All rights reserved.

1. Introduction

While spectral Fourier methods [1] provide the optimum efficiency for the high accuracy computation of regular solutions due to the spectral convergence and the availability of fast transformations, they require periodic boundary conditions or some particular treatment, such as damping zones, for more complex cases. Spectral methods based on more general orthogonal polynomials are also employed for flow simulations [2], but are often restricted to regular solutions and a reduced set of boundary conditions as well. Other high-order schemes like the discontinuous Galerkin (DG) methods or the spectral-element methods (SEM) have been conceived for geometrical flexibility and the possibility to adjust the order of approximation. For these (and other) reasons, the latter have been receiving vital interest from the community during the last years. Yet, fast solvers for these methods are still a matter of research.

As with high-order methods the number of degrees of freedom inside an element scales with the polynomial degree to the power of three, ways to reduce the algebraic problem size are sought. The static condensation method is often used to this end. For instance the first work on SEM in [3] already employs it, as do more recent ones [4,5]. Other applications of static condensation include the explicit solution for cuboidal geometries [6], p-multigrid techniques for rectangular geometries [7] and the application as preconditioner for a DG scheme [8].

The algorithms in the cited references benefit from the static condensation with spectacular increases in performance. However, they all share one downside: When increasing the polynomial degree, the operation count scales super-linearly

* Corresponding author at: Institute of Fluid Mechanics, TU Dresden, George-Bähr-Straße 3c, 01062 Dresden, Germany.
E-mail address: Immo.Huismann@tu-dresden.de (I. Huismann).

http://dx.doi.org/10.1016/j.jcp.2017.06.012
0021-9991/© 2017 Elsevier Inc. All rights reserved.

with the number of degrees of freedom, so that the method becomes less and less efficient with higher and higher polynomial degrees. To remain efficient at high polynomial degrees, linear complexity is required throughout the entire solver, from the operator execution, to the preconditioner, to the remaining operations inside an iteration.

As the implicit treatment of diffusion terms and pressure-velocity coupling in solvers for incompressible fluid flow often reduce to a Helmholtz equation, the goal of this article is to provide a Helmholtz solver with linear scaling. It extends earlier works of the authors in [9] and [10], where preliminary variants of the condensed Helmholtz operator with linear operation count were proposed. While these variants resulted in linear execution times of the iterations, they outperformed unfactorized versions implemented via dense matrix–matrix multiplications only for polynomial degrees $p > 10$. Current simulations, however, tend to use lower polynomial degrees [11,5,12] so that with these methods a gain might not be achieved. The present paper proposes an efficient static condensation method for a spectral-element discretization using cuboidal elements, outperforming matrix–matrix multiplications down to a polynomial degree of $p = 2$. Similar to the work in many of the cited references, the development is performed for a Cartesian discretization here.

The paper is laid out as follows: Section 2 focuses on tensor-product matrices as a necessary prerequisite and the third section on the spectral-element method. Section 4 recapitulates the static condensation while the fifth section recalls operators from [9]. Section 6, finally, puts these elements together and proposes the new method. In Section 7 and 8, the efficiency of the new operators and solvers is quantitatively assessed with suitable test cases.

2. Tensor-product matrices

Many partial differential equations exhibit a separable substructure [13], i.e. the differential operator can be decomposed into smaller operators acting in single coordinate directions only. This allows for further analysis of the operator and the resulting system matrices. Indeed, it is the basic approach to lower the operation count here, as illustrated by the following very simple example. Assume that a two-dimensional problem is discretized using a spectral method with n degrees of freedom in each direction such that the vector of discrete unknowns is $\mathbf{v} \in \mathbb{R}^{n^2}$. The system matrix $\mathbf{C} \in \mathbb{R}^{n^2,n^2}$ is dense, so that its straightforward application requires n^4 multiplications. If it is a tensor-product matrix, however, its application can be reformulated as

$$\mathbf{C}\mathbf{v} = (\mathbf{B} \otimes \mathbf{A})\,\mathbf{v} = (\mathbf{B} \otimes \mathbf{I})\,(\mathbf{I} \otimes \mathbf{A})\,\mathbf{v} \tag{1}$$

with $\mathbf{I} \in \mathbb{R}^{n,n}$ being the identity matrix, $\mathbf{A} \in \mathbb{R}^{n,n}$ the operator in the first direction and $\mathbf{B} \in \mathbb{R}^{n,n}$ the operator in the second one. The consecutive application of $\mathbf{A}$ and $\mathbf{B}$ then requires only $2n^3$ multiplications. In general, tensor products of dimension d require only dn^{d+1} multiplications compared to n^{2d} for the application of the whole matrix.

Tensor-product matrices possess additional properties that allow for factorization techniques. E.g., the multiplication of two tensor-product matrices is reducible to the multiplication of the respective submatrices

$$(\mathbf{A} \otimes \mathbf{C})\,(\mathbf{B} \otimes \mathbf{D}) = (\mathbf{A}\mathbf{B}) \otimes (\mathbf{C}\mathbf{D}) \quad . \tag{2}$$

Also, transpose and inverse can be reformulated

$$(\mathbf{A} \otimes \mathbf{B})^T = \mathbf{A}^T \otimes \mathbf{B}^T \tag{3}$$

$$(\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1} \quad . \tag{4}$$

Further properties and applications of tensor-product matrices are presented in [13] and [14].

3. The spectral-element method with cuboidal elements

The continuous Helmholtz problems in a domain Ω reads

$$\lambda u - \Delta u = f \quad , \tag{5}$$

where u is the variable to solve for, $\lambda \geq 0$ is a real constant parameter, Δ the Laplace operator and f the right-hand side. This equation was first formulated in the field of acoustic research with $\lambda < 0$. Nonetheless, the case $\lambda \geq 0$ is commonly referred to as Helmholtz equation in the fluid dynamics community.

Decomposing the domain into n_e elements Ω_e, the spectral-element method leads to the discrete equation system

$$\mathbf{R}\mathbf{H}_\mathrm{L}\mathbf{R}^T\mathbf{u}_\mathrm{G} = \mathbf{R}\mathbf{F}_\mathrm{L} \quad , \tag{6}$$

where $\mathbf{u}_\mathrm{G}$ is the solution vector, $\mathbf{F}_\mathrm{L}$ is the discretized right-hand side, $\mathbf{R}$ gathers the contributions from the elements, and its transpose $\mathbf{R}^T$ scatters the global degrees of freedom to those local to the elements [14,15]. The local Helmholtz operator $\mathbf{H}_\mathrm{L}$ is a block-diagonal matrix consisting of the element Helmholtz operators $\mathbf{H}_e$.

This paper only considers the case of cuboidal elements. A three-dimensional tensor-product basis is utilized in each element, allowing the standard element basis functions ϕ_{ijk} to be decomposed into three one-dimensional basis functions such that [14]

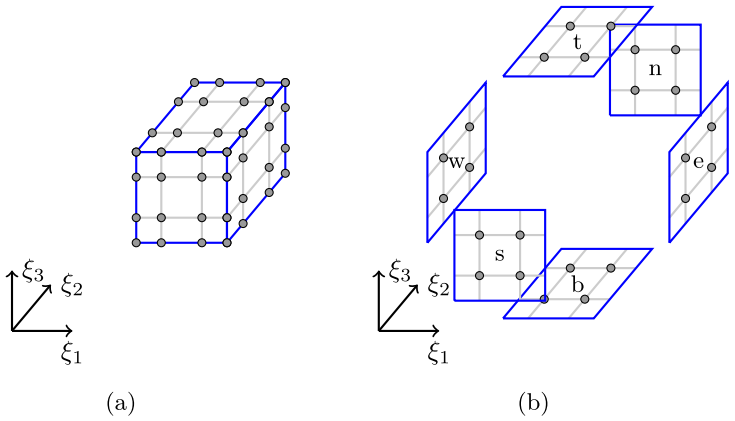

Fig. 1. Left: Three-dimensional tensor-product element with GLL nodes of order $p = 3$. Right: Compass notation for element faces.

$$\forall \quad 0 \leq i, j, k \leq p: \quad \phi_{ijk}(\xi_1, \xi_2, \xi_3) = \varphi_i(\xi_1)\,\varphi_j(\xi_2)\,\varphi_k(\xi_3) \tag{7}$$

with $\xi_1, \xi_2, \xi_3 \in [-1, 1]$ the local coordinates in the element and $\{\varphi_i : i = 0, \ldots, p\}$ the set of basis functions employed in all three directions. These one-dimensional functions result in the standard element mass and stiffness matrices

$$M_{ij} = \int_{-1}^{1} \varphi_i(\xi)\,\varphi_j(\xi)\,\mathrm{d}\xi \tag{8}$$

$$K_{ij} = \int_{-1}^{1} \varphi_i'(\xi)\,\varphi_j'(\xi)\,\mathrm{d}\xi \quad , \tag{9}$$

where the prime denotes differentiation. Using a tensor-product basis, the operations on each element can be decomposed and the Helmholtz operator becomes

$$\mathbf{H}_e = d_{e,0}\mathbf{M}\otimes\mathbf{M}\otimes\mathbf{M} + d_{e,1}\mathbf{M}\otimes\mathbf{M}\otimes\mathbf{K} + d_{e,2}\mathbf{M}\otimes\mathbf{K}\otimes\mathbf{M} + d_{e,3}\mathbf{K}\otimes\mathbf{M}\otimes\mathbf{M} \quad . \tag{10}$$

For an element Ω_e of extent $h_{e,i}$, $i \in \{1, 2, 3\}$, in the three coordinate directions the vector coefficients $\mathbf{d}_e$ in (10) reads

$$\mathbf{d}_e = \frac{h_{e,1}h_{e,2}h_{e,3}}{8}\left(\lambda \quad \frac{4}{h_{e,1}^2} \quad \frac{4}{h_{e,2}^2} \quad \frac{4}{h_{e,3}^2}\right)^T \quad . \tag{11}$$

For convenience, $d_{e,0}$ incorporates the Helmholtz parameter λ here, whereas the remaining components represent metric terms.

Throughout this paper, Lagrange polynomials to the Gauss–Lobatto–Legendre (GLL) quadrature points constitute the one-dimensional nodal basis functions. Hence, the tensor-product GLL points can be identified with the degrees of freedom of an element, as depicted in Fig. 1a. The mass matrix is approximated via GLL quadrature, generating a diagonal matrix and thereby simplifying the structure of the Helmholtz operator: The mass term $\mathbf{M}\otimes\mathbf{M}\otimes\mathbf{M}$ reduces to a diagonal matrix whereas each stiffness term becomes diagonal in two dimensions, e.g. $\mathbf{M}\otimes\mathbf{K}\otimes\mathbf{M}$ in direction one and three. While the simplification lowers the number of operations, it still scales with $\mathcal{O}(p^4)$ and, hence, super-linearly with respect to the number of unknowns. The super-linear complexity represents a roadblock on the path to higher polynomial degrees and reduces the efficiency of the solvers. Hence, the next sections focus on the removal of this obstruction.

4. The static condensation method

The Helmholtz equation is elliptic, so that the Dirichlet problem is well posed [16]. This property can be utilized to eliminate the internal nodes of an element, reducing the number of unknowns significantly, but coupling the remaining ones tighter. In the following, this will be discussed for a single element dropping the subscript e indicating the element.

The degrees of freedom of the element are sorted into those located on the boundary, denoted by the subscript B, and those in the interior, denoted by the subscript I, as illustrated in Fig. 2 for a two-dimensional element. The subscripts I and B are also used for the corresponding matrices, where applicable, so that, e.g., $\mathbf{H}_{\mathrm{IB}}$ stands for the part of the Helmholtz operator mapping from the boundary to the inner degrees of freedom, whereas $\mathbf{M}_{\mathrm{II}}$ refers to the inner part of the one-dimensional mass matrix. After sorting the variables accordingly, the element Helmholtz operator becomes

$$\begin{pmatrix} \mathbf{H}_{\mathrm{BB}} & \mathbf{H}_{\mathrm{BI}} \\ \mathbf{H}_{\mathrm{IB}} & \mathbf{H}_{\mathrm{II}} \end{pmatrix} \begin{pmatrix} \mathbf{u}_{\mathrm{B}} \\ \mathbf{u}_{\mathrm{I}} \end{pmatrix} = \begin{pmatrix} \mathbf{F}_{\mathrm{B}} \\ \mathbf{F}_{\mathrm{I}} \end{pmatrix} \quad . \tag{12}$$

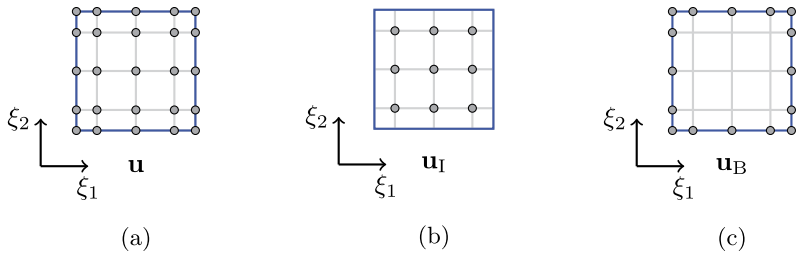

Fig. 2. Categorization of the degrees of freedom in a two-dimensional element with $p = 4$ into inner and boundary degrees. (a): All degrees of freedom in the element, (b): nodes corresponding to internal degrees of freedom, (c): only boundary nodes.

As $\mathbf{H}_\mathrm{II}$ corresponds to the homogeneous DIRICHLET operator and, hence, is invertible, $\mathbf{u}_\mathrm{I}$ equates to

$$\mathbf{u}_\mathrm{I} = \mathbf{H}_\mathrm{II}^{-1} \left(\mathbf{F}_\mathrm{I} - \mathbf{H}_\mathrm{IB} \mathbf{u}_\mathrm{B} \right) \quad . \tag{13}$$

This, in turn, yields

$$\underbrace{\left(\mathbf{H}_\mathrm{BB} - \mathbf{H}_\mathrm{BI} \mathbf{H}_\mathrm{II}^{-1} \mathbf{H}_\mathrm{IB} \right)}_{\hat{\mathbf{H}}} \mathbf{u}_\mathrm{B} = \underbrace{\mathbf{F}_\mathrm{B} - \mathbf{H}_\mathrm{BI} \mathbf{H}_\mathrm{II}^{-1} \mathbf{F}_\mathrm{I}}_{\hat{\mathbf{F}}} \quad . \tag{14}$$

The operator

$$\hat{\mathbf{H}} = \underbrace{\mathbf{H}_\mathrm{BB}}_{\mathbf{H}^\mathrm{prim}} - \underbrace{\mathbf{H}_\mathrm{BI} \mathbf{H}_\mathrm{II}^{-1} \mathbf{H}_\mathrm{IB}}_{\mathbf{H}^\mathrm{cond}} \tag{15}$$

defines a condensed element HELMHOLTZ operator with the rank equal to the number of boundary points. It is composed of two parts: The primary part, $\mathbf{H}^\mathrm{prim}$, is the restriction of the full HELMHOLTZ operator to the boundary nodes, whereas the condensed part, $\mathbf{H}^\mathrm{cond}$, stems from the condensation process and represents the interaction of the boundary values and the internal degrees of freedom.

After solving the equation system for the element boundary values (14), equation (13) defines the interior unknowns. Algorithm 1 summarizes the resulting solution process.

Algorithm 1 Solution algorithm with static condensation.

Restrict to boundary nodes, condense right-hand side
foreach Ω_e **do**
  $\hat{\mathbf{F}}_e \leftarrow \mathbf{F}_{\mathrm{B},e} - \mathbf{H}_{\mathrm{BI},e} \mathbf{H}_{\mathrm{II},e}^{-1} \mathbf{F}_{\mathrm{I},e}$
  $\hat{\mathbf{u}}_e \leftarrow \mathbf{u}_{\mathrm{B},e}$
end

$\hat{\mathbf{u}} \leftarrow \mathrm{Solution}(\hat{\mathbf{R}} \hat{\mathbf{H}} \hat{\mathbf{R}}^T \hat{\mathbf{u}} = \hat{\mathbf{R}} \hat{\mathbf{F}})$

Regain interior degrees of freedom
foreach Ω_e **do**
  $\mathbf{u}_{\mathrm{I},e} \leftarrow \mathbf{H}_{\mathrm{II},e}^{-1} \left(\mathbf{F}_{\mathrm{I},e} - \mathbf{H}_{\mathrm{IB},e} \mathbf{u}_{\mathrm{B},e} \right)$
  $\mathbf{u}_e \leftarrow \left(\mathbf{u}_{\mathrm{B},e} \quad \mathbf{u}_{\mathrm{I},e} \right)$
end

Attention to detail is required in the implementation of $\hat{\mathbf{H}}$. A naive matrix–matrix implementation requires $\mathcal{O}\left(p^4\right)$ multiplications and, hence, scales super-linearly with the number of degrees of freedom. In contrast, treating the primary and condensed part separately and exploiting the tensor-product structure yields linear scaling, as shown in [10].

5. Sum-factorization of the condensed equation

The condensed operator $\hat{\mathbf{H}}$ is composed of the primary and the condensed part: The former is the restriction of the full HELMHOLTZ operator to the boundary nodes and the suboperators inherit the tensor-product structure. Since most of the remaining degrees of freedom reside on the element faces, the face-to-face coupling is the most expensive suboperator and requires investigation. In the following compass notation will be utilized to identify faces of the element by the index set $\mathcal{I} = \{\mathrm{w}, \mathrm{e}, \mathrm{s}, \mathrm{n}, \mathrm{b}, \mathrm{t}\}$. As all of the suboperators for the six faces are similar, the discussion will utilize the east face, $\mathrm{F_e}$ in compass notation (Fig. 1b). The degrees of freedom on this face, $\mathbf{u}_{\mathrm{F_e}}$, correspond to $\left\{u_{ijp}\right\}_{i,j=1,\ldots,p-1}$ using the notation from (7). Hence, the primary part from the east face onto itself can be written as

$$\mathbf{H}_{\mathrm{F_eF_e}} = d_0\mathbf{M}_{\mathrm{II}} \otimes \mathbf{M}_{\mathrm{II}} \otimes M_{pp} + d_1\mathbf{M}_{\mathrm{II}} \otimes \mathbf{M}_{\mathrm{II}} \otimes K_{pp} + d_2\mathbf{M}_{\mathrm{II}} \otimes \mathbf{K}_{\mathrm{II}} \otimes M_{pp} + d_3\mathbf{K}_{\mathrm{II}} \otimes \mathbf{M}_{\mathrm{II}} \otimes M_{pp} \quad .$$

Due to the symmetry of the system matrices and the last dimension of the tensor-product operating on only one value, the above becomes

$$\mathbf{H}_{\mathrm{F_eF_e}} = d_0 M_{00}\mathbf{M}_{\mathrm{II}} \otimes \mathbf{M}_{\mathrm{II}} + d_1 K_{00}\mathbf{M}_{\mathrm{II}} \otimes \mathbf{M}_{\mathrm{II}} + d_2 M_{00}\mathbf{M}_{\mathrm{II}} \otimes \mathbf{K}_{\mathrm{II}} + d_3 M_{00}\mathbf{K}_{\mathrm{II}} \otimes \mathbf{M}_{\mathrm{II}} \quad , \tag{16}$$

with $d_0 \dots d_3$ from (11) and $\mathbf{M}_{\mathrm{II}}$ and $\mathbf{K}_{\mathrm{II}}$ as the interior part of the one-dimensional mass and stiffness matrices. This expression is readily evaluated in $2n_{\mathrm{I}}^3 + \mathcal{O}\left(n_{\mathrm{I}}^2\right)$ multiplications, where the number of points per face is $n_{\mathrm{I}}^2 = (p-1)^2$. The other terms of the primary part map between opposite faces and are diagonal, e.g.

$$\mathbf{H}_{\mathrm{F_wF_e}} = d_1 K_{0p}\mathbf{M}_{\mathrm{II}} \otimes \mathbf{M}_{\mathrm{II}} \quad . \tag{17}$$

Due to the diagonal mass matrix only the Laplace terms couple collocation points, and a face gets only coupled with the edges around it and the opposite face. The cross-terms between non-opposing faces, e.g. east and north face, compute to zero. Hence, the whole primary part requires $6 \cdot 2n_{\mathrm{I}}^3 + \mathcal{O}\left(n_{\mathrm{I}}^2\right)$ multiplications and scales linearly with the number of degrees of freedom.

The condensed part consists of three suboperators:

$$\mathbf{H}^{\mathrm{cond}} = \mathbf{H}_{\mathrm{BI}}\mathbf{H}_{\mathrm{II}}^{-1}\mathbf{H}_{\mathrm{IB}} \quad .$$

As established in Section 3, only the Laplace terms infer interaction between nodes in an element. This interaction is along mesh lines. Hence, the element edges and vertices do not map into the interior, only the faces of the element do. A first implementation of the condensed part consists of precomputing the face-to-face operators and using them directly, as done in Algorithm 2. Since the matrices possess $n_{\mathrm{I}}^2 \cdot n_{\mathrm{I}}^2$ entries, the algorithm requires $6n_{\mathrm{I}}^2 \cdot 6n_{\mathrm{I}}^2 = 36n_{\mathrm{I}}^4$ multiplications. The scaling is super-linear with respect to the number of degrees of freedom, but an implementation can benefit from optimized libraries, e.g. level 3 BLAS routines [17], mitigating this drawback.

Algorithm 2 Evaluation of the condensed operator in a direct face-to-face variant.

foreach $j \in \mathcal{I}$ **do**
  $\mathbf{v}_{\mathrm{F}_j} \leftarrow \sum_{i\in\mathcal{I}} \mathbf{H}^{\mathrm{cond}}_{\mathrm{F}_j\mathrm{F}_i}\mathbf{u}_{\mathrm{F}_i}$
end

Reference [9] of the present authors investigated a linearly scaling variant of the operator based on tensor products. Linear scaling is possible when exploiting the tensor-product structure of all suboperators. For $\mathbf{H}_{\mathrm{BI}}$ and $\mathbf{H}_{\mathrm{IB}}$ a tensor-product structure is induced by the restriction of (10) to the boundary nodes. The inverse $\mathbf{H}_{\mathrm{II}}^{-1}$ can be expressed via the fast diagonalization method [13] to yield

$$\mathbf{H}_{\mathrm{II}}^{-1} = \left(\mathbf{S}_{\mathrm{II}} \otimes \mathbf{S}_{\mathrm{II}} \otimes \mathbf{S}_{\mathrm{II}}\right)\mathbf{D}^{-1}\left(\mathbf{S}_{\mathrm{II}} \otimes \mathbf{S}_{\mathrm{II}} \otimes \mathbf{S}_{\mathrm{II}}\right)^T \tag{18}$$

where

$$\mathbf{D} = d_0\mathbf{I} \otimes \mathbf{I} \otimes \mathbf{I} + d_1\mathbf{I} \otimes \mathbf{I} \otimes \mathbf{\Lambda} + d_2\mathbf{I} \otimes \mathbf{\Lambda} \otimes \mathbf{I} + d_3\mathbf{\Lambda} \otimes \mathbf{I} \otimes \mathbf{I} \tag{19}$$

and

$$\mathbf{S}_{\mathrm{II}}\mathbf{M}_{\mathrm{II}}\mathbf{S}_{\mathrm{II}}^T = \mathbf{I}, \quad \mathbf{S}_{\mathrm{II}}\mathbf{K}_{\mathrm{II}}\mathbf{S}_{\mathrm{II}}^T = \mathbf{\Lambda} \quad . \tag{20}$$

The last expression defines the one-dimensional transformation matrix $\mathbf{S}_{\mathrm{II}}$ and the matrix of eigenvalues $\mathbf{\Lambda}$ to the generalized eigenproblem for $\mathbf{K}_{\mathrm{II}}$ and $\mathbf{M}_{\mathrm{II}}$. While the diagonal matrix $\mathbf{D}$ is constructed of tensor-product matrices, its inverse is diagonal as well but not a tensor-product matrix any more. Instead of using Algorithm 2, a sum-factorization in the inner element eigenspace can be utilized for the face-to-face operators, e.g. for face west

$$\begin{aligned}
\mathbf{v}_{\mathrm{F_w}} &= \sum_{i\in\mathcal{I}} \mathbf{H}^{\mathrm{cond}}_{\mathrm{F_wF}_i}\mathbf{u}_{\mathrm{F}_i} \\
&= \sum_{i\in\mathcal{I}} \underbrace{\mathbf{H}_{\mathrm{F_wI}}\left(\mathbf{S}_{\mathrm{II}} \otimes \mathbf{S}_{\mathrm{II}} \otimes \mathbf{S}_{\mathrm{II}}\right)}_{\mathbf{H}_{\mathrm{F_wE}}}\mathbf{D}^{-1}\underbrace{\left(\mathbf{S}_{\mathrm{II}}^T \otimes \mathbf{S}_{\mathrm{II}}^T \otimes \mathbf{S}_{\mathrm{II}}^T\right)\mathbf{H}_{\mathrm{IF}_i}}_{\mathbf{H}_{\mathrm{EF}_i}}\mathbf{u}_{\mathrm{F}_i} \\
\Leftrightarrow \mathbf{v}_{\mathrm{F_w}} &= \mathbf{H}_{\mathrm{F_wE}}\mathbf{D}^{-1}\sum_{i\in\mathcal{I}} \mathbf{H}_{\mathrm{EF}_i}\mathbf{u}_{\mathrm{F}_i} \quad ,
\end{aligned} \tag{21}$$

where the index E denotes the inner element eigenspace. Table 1 lists all the suboperators $\mathbf{H}_{\mathrm{EF}_i}$. Being products of tensor-product matrices they are tensor-product matrices themselves and are applicable in $3n_{\mathrm{I}}^3$ multiplications. First computing $\mathbf{D}^{-1}\sum_{i\in\mathcal{I}} \mathbf{H}_{\mathrm{EF}_i}\mathbf{u}_{\mathrm{F}_i}$, then using it to calculate the six vectors $\mathbf{v}_{\mathrm{F}_j}$ leads to Algorithm 3. This algorithm evaluates the

Table 1
Suboperators of the factorized condensed part of one element.

i	$\mathbf{H}_{\mathrm{F}_i\mathrm{E}}$	$\mathbf{H}_{\mathrm{EF}_i}$
w	$d_1\,(\mathbf{M}_{\mathrm{II}}\mathbf{S}_{\mathrm{II}})\otimes(\mathbf{M}_{\mathrm{II}}\mathbf{S}_{\mathrm{II}})\otimes(\mathbf{K}_{0\mathrm{I}}\mathbf{S}_{\mathrm{II}})$	$d_1\,(\mathbf{S}_{\mathrm{II}}^T\mathbf{M}_{\mathrm{II}})\otimes(\mathbf{S}_{\mathrm{II}}^T\mathbf{M}_{\mathrm{II}})\otimes(\mathbf{S}_{\mathrm{II}}^T\mathbf{K}_{\mathrm{I}0})$
e	$d_1\,(\mathbf{M}_{\mathrm{II}}\mathbf{S}_{\mathrm{II}})\otimes(\mathbf{M}_{\mathrm{II}}\mathbf{S}_{\mathrm{II}})\otimes(\mathbf{K}_{p\mathrm{I}}\mathbf{S}_{\mathrm{II}})$	$d_1\,(\mathbf{S}_{\mathrm{II}}^T\mathbf{M}_{\mathrm{II}})\otimes(\mathbf{S}_{\mathrm{II}}^T\mathbf{M}_{\mathrm{II}})\otimes(\mathbf{S}_{\mathrm{II}}^T\mathbf{K}_{\mathrm{I}p})$
s	$d_2\,(\mathbf{M}_{\mathrm{II}}\mathbf{S}_{\mathrm{II}})\otimes(\mathbf{K}_{0\mathrm{I}}\mathbf{S}_{\mathrm{II}})\otimes(\mathbf{M}_{\mathrm{II}}\mathbf{S}_{\mathrm{II}})$	$d_2\,(\mathbf{S}_{\mathrm{II}}^T\mathbf{M}_{\mathrm{II}})\otimes(\mathbf{S}_{\mathrm{II}}^T\mathbf{K}_{\mathrm{I}0})\otimes(\mathbf{S}_{\mathrm{II}}^T\mathbf{M}_{\mathrm{II}})$
n	$d_2\,(\mathbf{M}_{\mathrm{II}}\mathbf{S}_{\mathrm{II}})\otimes(\mathbf{K}_{p\mathrm{I}}\mathbf{S}_{\mathrm{II}})\otimes(\mathbf{M}_{\mathrm{II}}\mathbf{S}_{\mathrm{II}})$	$d_2\,(\mathbf{S}_{\mathrm{II}}^T\mathbf{M}_{\mathrm{II}})\otimes(\mathbf{S}_{\mathrm{II}}^T\mathbf{K}_{\mathrm{I}p})\otimes(\mathbf{S}_{\mathrm{II}}^T\mathbf{M}_{\mathrm{II}})$
b	$d_3\,(\mathbf{K}_{0\mathrm{I}}\mathbf{S}_{\mathrm{II}})\otimes(\mathbf{M}_{\mathrm{II}}\mathbf{S}_{\mathrm{II}})\otimes(\mathbf{M}_{\mathrm{II}}\mathbf{S}_{\mathrm{II}})$	$d_3\,(\mathbf{S}_{\mathrm{II}}^T\mathbf{K}_{\mathrm{I}0})\otimes(\mathbf{S}_{\mathrm{II}}^T\mathbf{M}_{\mathrm{II}})\otimes(\mathbf{S}_{\mathrm{II}}^T\mathbf{M}_{\mathrm{II}})$
t	$d_3\,(\mathbf{K}_{p\mathrm{I}}\mathbf{S}_{\mathrm{II}})\otimes(\mathbf{M}_{\mathrm{II}}\mathbf{S}_{\mathrm{II}})\otimes(\mathbf{M}_{\mathrm{II}}\mathbf{S}_{\mathrm{II}})$	$d_3\,(\mathbf{S}_{\mathrm{II}}^T\mathbf{K}_{\mathrm{I}p})\otimes(\mathbf{S}_{\mathrm{II}}^T\mathbf{M}_{\mathrm{II}})\otimes(\mathbf{S}_{\mathrm{II}}^T\mathbf{M}_{\mathrm{II}})$

condensed part in $37n_{\mathrm{I}}^3$ multiplications and, thus, achieves linear complexity. Additionally, the memory requirements become linear as well, since only $\mathbf{D}^{-1}$ is required, whereas all the other matrices are independent of the number of elements.

While Algorithm 3 uses fewer multiplications for $p \geq 3$, the implementation was only faster for polynomial degrees $p > 10$ [9]. As current high-order large-scale simulations employ polynomial degrees ranging between 8 and 12 [11, 18], the possible gains actually achieved are small or even negative. However, further factorization leads to more efficient operators, as developed below.

Algorithm 3 Evaluation of the condensed part that accumulates contributions in the eigenspace and then maps back to the faces.

$\check{\mathbf{u}} \leftarrow \sum_{i\in\mathcal{I}} \mathbf{H}_{\mathrm{EF}_i}\mathbf{u}_{\mathrm{F}_i}$
$\check{\mathbf{v}} \leftarrow \mathbf{D}^{-1}\check{\mathbf{u}}$
foreach $j \in \mathcal{I}$ **do**
  $\mathbf{v}_{\mathrm{F}_j} \leftarrow \mathbf{H}_{\mathrm{F}_j\mathrm{E}}\check{\mathbf{v}}$
end

6. Factorizing the factorization

Table 1 assembles the tensor-product suboperators of the condensed part used in Algorithm 3. Two thirds of the matrix operations stem from the application of $\mathbf{M}_{\mathrm{II}}\mathbf{S}_{\mathrm{II}}$ or its transpose. Eliminating these common terms lowers the multiplication count considerably and is a key to achieving better performance. A coordinate transformation provides an easy approach towards this goal, as it leads to new system matrices $\widetilde{\mathbf{K}}$ and $\widetilde{\mathbf{M}}$, which possess favorable properties. By applying the matrix

$$\mathbf{S} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \mathbf{S}_{\mathrm{II}} & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{22}$$

to all three directions, the element Helmholtz operator $\mathbf{H}_e$ defined in (10) is transformed to

$$\begin{aligned} \widetilde{\mathbf{H}}_e &:= (\mathbf{S}\otimes\mathbf{S}\otimes\mathbf{S})\,\mathbf{H}_e\left(\mathbf{S}^T\otimes\mathbf{S}^T\otimes\mathbf{S}^T\right) \\ &= d_{e,0}\left(\mathbf{S}\mathbf{M}\mathbf{S}^T\right)\otimes\left(\mathbf{S}\mathbf{M}\mathbf{S}^T\right)\otimes\left(\mathbf{S}\mathbf{M}\mathbf{S}^T\right) + d_{e,1}\left(\mathbf{S}\mathbf{M}\mathbf{S}^T\right)\otimes\left(\mathbf{S}\mathbf{M}\mathbf{S}^T\right)\otimes\left(\mathbf{S}\mathbf{K}\mathbf{S}^T\right) \\ &\quad + d_{e,2}\left(\mathbf{S}\mathbf{M}\mathbf{S}^T\right)\otimes\left(\mathbf{S}\mathbf{K}\mathbf{S}^T\right)\otimes\left(\mathbf{S}\mathbf{M}\mathbf{S}^T\right) + d_{e,3}\left(\mathbf{S}\mathbf{K}\mathbf{S}^T\right)\otimes\left(\mathbf{S}\mathbf{M}\mathbf{S}^T\right)\otimes\left(\mathbf{S}\mathbf{K}\mathbf{S}^T\right) \quad . \end{aligned} \tag{23}$$

Defining the transformed mass matrix

$$\widetilde{\mathbf{M}} = \mathbf{S}\mathbf{M}\mathbf{S}^T = \begin{pmatrix} \mathbf{M}_{00} & 0 & 0 \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & \mathbf{M}_{pp} \end{pmatrix} \tag{24}$$

and the transformed stiffness matrix,

$$\widetilde{\mathbf{K}} = \mathbf{S}\mathbf{K}\mathbf{S}^T = \begin{pmatrix} \mathbf{K}_{00} & \mathbf{K}_{0\mathrm{I}}\mathbf{S}_{\mathrm{II}}^T & \mathbf{K}_{0p} \\ \mathbf{S}_{\mathrm{II}}\mathbf{K}_{\mathrm{I}0} & \mathbf{\Lambda} & \mathbf{S}_{\mathrm{II}}\mathbf{K}_{\mathrm{I}p} \\ \mathbf{K}_{p0} & \mathbf{K}_{p\mathrm{I}}\mathbf{S}_{\mathrm{II}}^T & \mathbf{K}_{pp} \end{pmatrix} \quad , \tag{25}$$

reduces the transformed Helmholtz operator to

$$\widetilde{\mathbf{H}}_e = d_{e,0}\widetilde{\mathbf{M}}\otimes\widetilde{\mathbf{M}}\otimes\widetilde{\mathbf{M}} + d_{e,1}\widetilde{\mathbf{M}}\otimes\widetilde{\mathbf{M}}\otimes\widetilde{\mathbf{K}} + d_{e,2}\widetilde{\mathbf{M}}\otimes\widetilde{\mathbf{K}}\otimes\widetilde{\mathbf{M}} + d_{e,3}\widetilde{\mathbf{K}}\otimes\widetilde{\mathbf{M}}\otimes\widetilde{\mathbf{M}} \quad . \tag{26}$$

In the transformed system both $\widetilde{\mathbf{M}}_{\mathrm{II}} = \mathbf{I}$ and $\widetilde{\mathbf{K}}_{\mathrm{II}} = \mathbf{\Lambda}$ are diagonal. As a result, the generalized eigenvalue decomposition of $\widetilde{\mathbf{K}}_{\mathrm{II}}$ with respect to $\widetilde{\mathbf{M}}_{\mathrm{II}} = \mathbf{I}$ possesses the transformation matrix

Table 2
Suboperators of the factorized condensed part in the transformed system.

i	$\widetilde{\mathbf{H}}_{\mathrm{F}_i\mathrm{E}}$	$\widetilde{\mathbf{H}}_{\mathrm{EF}_i}$
w	$d_1\mathbf{I}\otimes\mathbf{I}\otimes\widetilde{\mathbf{K}}_{0\mathrm{I}}$	$d_1\mathbf{I}\otimes\mathbf{I}\otimes\widetilde{\mathbf{K}}_{\mathrm{I}0}$
e	$d_1\mathbf{I}\otimes\mathbf{I}\otimes\widetilde{\mathbf{K}}_{p\mathrm{I}}$	$d_1\mathbf{I}\otimes\mathbf{I}\otimes\widetilde{\mathbf{K}}_{\mathrm{I}p}$
s	$d_2\mathbf{I}\otimes\widetilde{\mathbf{K}}_{0\mathrm{I}}\otimes\mathbf{I}$	$d_2\mathbf{I}\otimes\widetilde{\mathbf{K}}_{\mathrm{I}0}\otimes\mathbf{I}$
n	$d_2\mathbf{I}\otimes\widetilde{\mathbf{K}}_{p\mathrm{I}}\otimes\mathbf{I}$	$d_2\mathbf{I}\otimes\widetilde{\mathbf{K}}_{\mathrm{I}p}\otimes\mathbf{I}$
b	$d_3\widetilde{\mathbf{K}}_{0\mathrm{I}}\otimes\mathbf{I}\otimes\mathbf{I}$	$d_3\widetilde{\mathbf{K}}_{\mathrm{I}0}\otimes\mathbf{I}\otimes\mathbf{I}$
t	$d_3\widetilde{\mathbf{K}}_{p\mathrm{I}}\otimes\mathbf{I}\otimes\mathbf{I}$	$d_3\widetilde{\mathbf{K}}_{\mathrm{I}p}\otimes\mathbf{I}\otimes\mathbf{I}$

$$\widetilde{\mathbf{S}}_{\mathrm{II}} = \mathbf{I} \tag{27}$$

$$\Rightarrow \widetilde{\mathbf{M}}_{\mathrm{II}}\widetilde{\mathbf{S}}_{\mathrm{II}} = \mathbf{I} \quad . \tag{28}$$

The above identities simplify the suboperators from Table 1 to those in Table 2, thereby lowering the operation count. Where the condensed part of the original operator required $13n_\mathrm{I}^3 + 24n_\mathrm{I}^3$ multiplications, the condensed part of the transformed system utilizes only $13n_\mathrm{I}^3$ multiplications only. In addition, the primary part simplifies as well. The matrix $\mathbf{H}_{\mathrm{F_eF_e}}$ in (16), e.g., becomes

$$\widetilde{\mathbf{H}}_{\mathrm{F_eF_e}} = d_0M_{00}\mathbf{I}\otimes\mathbf{I} + d_1K_{00}\mathbf{I}\otimes\mathbf{I} + d_2M_{00}\mathbf{I}\otimes\mathbf{\Lambda} + d_3M_{00}\mathbf{\Lambda}\otimes\mathbf{I} \tag{29}$$

which is diagonal as well. The primary part of the transformed system now requires $\mathcal{O}\left(n_\mathrm{I}^2\right)$ multiplications compared to $12n_\mathrm{I}^3 + \mathcal{O}\left(n_\mathrm{I}^2\right)$ with the original form (16).

While the operator application simplifies, the pre- and post-processing steps expand due to the transformation. This is reflected by Algorithm 4, which solves the transformed system involving the transformed operator $\hat{\widetilde{\mathbf{H}}} = \widetilde{\mathbf{H}}_{\mathrm{BB}} - \widetilde{\mathbf{H}}_{\mathrm{BI}}\widetilde{\mathbf{H}}_{\mathrm{II}}^{-1}\widetilde{\mathbf{H}}_{\mathrm{IB}}$.

Algorithm 4 Solution algorithm with static condensation in transformed system.

Transform, restrict to boundary nodes, condense right-hand side
foreach Ω_e **do**
  $\widetilde{\mathbf{u}}_e \leftarrow \left(\mathbf{S}^{-1}\otimes\mathbf{S}^{-1}\otimes\mathbf{S}^{-1}\right)^T\mathbf{u}_e$
  $\widetilde{\mathbf{F}}_e \leftarrow \left(\mathbf{S}^{-1}\otimes\mathbf{S}^{-1}\otimes\mathbf{S}^{-1}\right)\mathbf{F}_e$
  $\hat{\widetilde{\mathbf{F}}}_e \leftarrow \widetilde{\mathbf{F}}_{\mathrm{B},e} - \widetilde{\mathbf{H}}_{\mathrm{BI},e}\widetilde{\mathbf{H}}_{\mathrm{II},e}^{-1}\widetilde{\mathbf{F}}_{\mathrm{I},e}$
  $\hat{\widetilde{\mathbf{u}}}_e \leftarrow \widetilde{\mathbf{u}}_{\mathrm{B},e}$
end

$\hat{\widetilde{\mathbf{u}}} \leftarrow \mathrm{Solution}(\hat{\mathbf{R}}\hat{\widetilde{\mathbf{H}}}\hat{\mathbf{R}}^T\hat{\widetilde{\mathbf{u}}} = \hat{\mathbf{R}}\hat{\widetilde{\mathbf{F}}})$

Regain interior degrees of freedom, transform back
foreach Ω_e **do**
  $\widetilde{\mathbf{u}}_{\mathrm{I},e} \leftarrow \widetilde{\mathbf{H}}_{\mathrm{II},e}^{-1}\left(\widetilde{\mathbf{F}}_{\mathrm{I},e} - \widetilde{\mathbf{H}}_{\mathrm{IB},e}\widetilde{\mathbf{u}}_{\mathrm{B},e}\right)$
  $\widetilde{\mathbf{u}}_e \leftarrow \left(\widetilde{\mathbf{u}}_{\mathrm{B},e} \quad \widetilde{\mathbf{u}}_{\mathrm{I},e}\right)$
  $\mathbf{u}_e \leftarrow \left(\mathbf{S}\otimes\mathbf{S}\otimes\mathbf{S}\right)^T\widetilde{\mathbf{u}}_e$
end

7. Efficiency of operators

The previous sections presented several variants to apply the condensed Helmholtz operator. The first one realizes Algorithm 2, the naive approach, using a single full matrix–matrix multiplication to couple the faces of the condensed element. It is labeled MMC. The matrix incorporates a primary and a condensed part and requires $36n_\mathrm{I}^4n_\mathrm{e}$ multiplications for application. The second variant implements Algorithm 3 with tensor products and is labeled TPC. It uses $12n_\mathrm{I}^3n_\mathrm{e}$ multiplications for the primary and $37n_\mathrm{I}^3n_\mathrm{e}$ for the condensed part. The tensor-product variant for the transformed system is termed TPT in the following and only requires $13n_\mathrm{I}^3n_\mathrm{e}$ multiplications in total. Table 3 summarizes the multiplication count of the variants and their precomputation costs.

Runtime is not proportional to the number of multiplications, as loading, storing, and execution take time as well. Hence, performance tests were conducted to directly measure the efficiency of the different operators. The polynomial degree was varied between $2 \leq p \leq 32$ for a constant number of elements $n_\mathrm{e} = 512$ and a Helmholtz parameter $\lambda = \pi$.

The operators were implemented in Fortran 2008 and compiled with the Intel Fortran compiler v. 2015, where a single call to DGEMM served for computing the face-to-face interaction in MMC. The measurements were conducted on a single core of an Intel Xeon E5-2690. The setup and application of each operator was repeated 101 times and the last 100 times averaged, to remove effects from library loading.

Table 3
Complexities of leading terms of application and precomputation steps for three different variants of the condensed Helmholtz operator.

Variant	Precomputation	Primary part	Condensed part
MMC	$\mathcal{O}\left(n_\mathrm{l}^5 n_\mathrm{e}\right)$	$56 n_\mathrm{l}^2 n_\mathrm{e}$	$36 n_\mathrm{l}^4 n_\mathrm{e}$
TPC	$\mathcal{O}\left(n_\mathrm{l}^3 n_\mathrm{e}\right)$	$12 n_\mathrm{l}^3 n_\mathrm{e}$	$37 n_\mathrm{l}^3 n_\mathrm{e}$
TPT	$\mathcal{O}\left(n_\mathrm{l}^3 n_\mathrm{e}\right)$	$68 n_\mathrm{l}^2 n_\mathrm{e}$	$13 n_\mathrm{l}^3 n_\mathrm{e}$

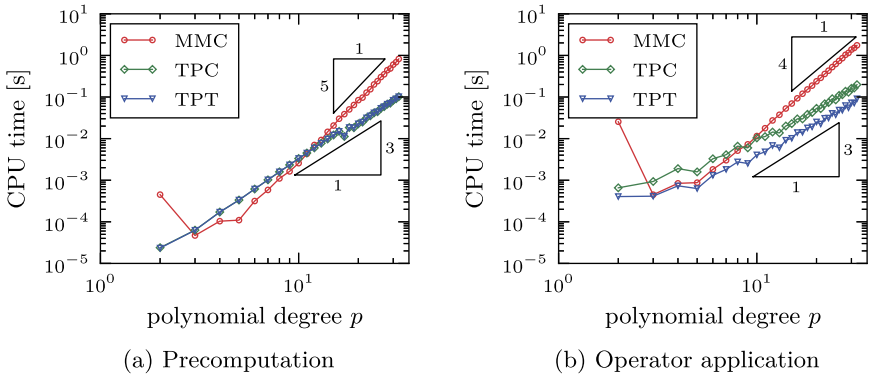

Fig. 3. Average runtime of matrix precomputation and operator evaluation with the different variants as a function of the polynomial degree p using $n_\mathrm{e} = 512$. (a): Operator precomputation, (b): operator application.

Fig. 3a depicts operator setup times. The precomputation scales with the expected $\mathcal{O}\left(n_\mathrm{l}^5\right)$ for MMC [4] and with $\mathcal{O}\left(n_\mathrm{l}^3\right)$ for TPC and TPT. For homogeneous grids, where the setup of MMC reduces to that of one element, the setup times are in the same order of magnitude as the time for a single application of the operator, shown in Fig. 3b, and can therefore be neglected for unsteady solution processes requiring a huge number of time steps. With non-uniform meshes, the setup time of MMC increases n_e-fold, possibly dominating the runtime of the whole solver process, as the setup time then is equal to the time of about 200 operator applications. Furthermore, storing the full matrices becomes a problem in particular with double precision. Using a polynomial degree of $p = 17$ and $n_\mathrm{e} = 512$ the face-to-face matrices require approximately 9.7 Gigabyte of memory. The variants TPC and TPT do not encounter these problems due to the linear scaling of their memory requirements.

Fig. 3b depicts measured operator execution times. For MMC the operator runtime starts with some oddity at $p = 2$, but scales with n_l^4 starting from $p = 5$. TPC starts with a higher runtime for $p = 3$, but due to the lower slope it becomes faster than MMC at $p = 10$ and achieves a speedup of 10 over MMC for $p = 32$. TPT exhibits the best of both worlds: It starts with a lower runtime than MMC and scales as TPC does. Furthermore, it is faster than TPC by a factor of more than 2 and is 20 times faster than MMC for $p = 32$.

According to Table 3, a slope of 3 is expected for the tensor-product based versions, but the measured runtimes exhibit a slightly smaller slope of about 2.8. Multiple explanations are possible. First, and foremost, the primary part consists of many suboperators whose operation count scales with $\mathcal{O}\left(n_\mathrm{l}^2 n_\mathrm{e}\right)$. Second, the implementation consists of loops with n_l iterations. These become more efficient as the polynomial degree increases. The combination of both can yield the lower slope. The result is an operator whose execution time scales sub-linearly with respect to the number of degrees of freedom.

8. Performance of solvers

The previous sections focused on the linear scaling of operators with the number of degrees of freedom when changing the order as a prerequisite for solvers with linear scaling. A typical solver, however, does not only consist of the operator to be applied. A good iteration scheme and preconditioner are required as well for the fast solution of the given equation. In most cases, multigrid techniques are employed to achieve a constant iteration count. To investigate the impact of the condition of the system matrices on the solution procedure, a conjugate gradient solver suffices [19,20], leaving only the choice of the preconditioner to be made.

As the preconditioner is required to scale linearly with respect to the number of degrees of freedom, only diagonal and tensor-product preconditioners are suited. In [9], block-Jacobi preconditioners for the faces of the elements were investigated. These employ the exact inverse of the operators from a face to itself and can be evaluated in tensor-product form. The remaining grid entities, i.e. edges and vertices, are treated similarly. This preconditioner is referred to as the block preconditioner in the following.

Four solvers are tested here. The first one, labeled UC, is an unpreconditioned solver for the condensed system. The second one, DC, adds diagonal preconditioning. The third one, BC, applies block preconditioning to the condensed system.

Table 4
Complexities of the leading terms of the different parts of the solvers investigated.

Solver	Variant	Operator	Preconditioning	Preconditioner
UC	TPC	$49n_l^3 n_e$	None	–
DC	TPC	$49n_l^3 n_e$	Diagonal	$6n_l^2 n_e$
BC	TPC	$49n_l^3 n_e$	Block	$24n_l^3 n_e$
BT	TPT	$13n_l^3 n_e$	Diagonal	$6n_l^2 n_e$

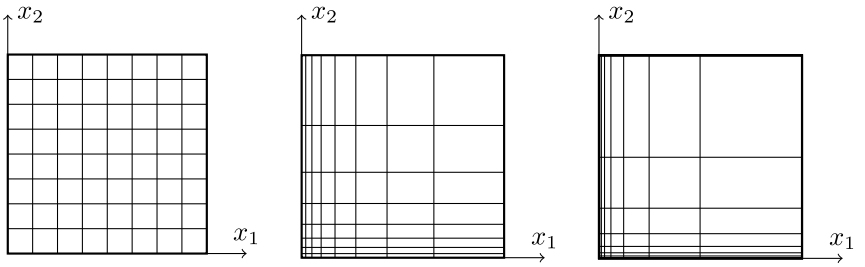

Fig. 4. Meshes with 8×8 elements in the x_1–x_2-plane for different values of the expansion factor.

All three variants utilize TPC as evaluation method for the static condensed Helmholtz operator. The fourth solver, labeled BT, works in the transformed system and applies TPT in combination with block preconditioning, which reduces to the application of a diagonal matrix in the transformed system. Table 4 summarizes the complexities of one iteration.

The test problem considered is created by a manufactured solution to (5) in the domain $\Omega = (0, 2\pi)^3$ with inhomogeneous Dirichlet boundary conditions. The chosen solution is

$$u_{ex}(x) = \cos(k(x_1 - 3x_2 + 2x_3)) \sin(k(1 + x_1)) \cdot \sin(k(1 - x_2)) \sin(k(2x_1 + x_2)) \sin(k(3x_1 - 2x_2 + 2x_3)) \quad , \tag{30}$$

generalizing the one employed in [7] to three dimensions. The right-hand side of (5) is evaluated analytically from

$$f(x) = \lambda u_{ex}(x) - \Delta u_{ex}(x) \quad , \tag{31}$$

and the boundary conditions set to u_{ex}.

In the following, the case $\lambda = 0$ is investigated. This in fact is the Poisson equation for which the resulting system matrix is not diagonally dominant anymore. Hence, this case is harder than $\lambda > 0$, thus providing the ultimate test. The stiffness parameter in (30) is set to $k = 5$.

The domain is discretized using $n_e = 8 \times 8 \times 8$ elements, where a constant expansion factor α between the elements leads to a non-uniform spacing as illustrated in Fig. 4. As a result, the aspect ratio of the elements in the grids can differ substantially from element to element when α gets larger. This leads to elements of cube-like, pancake-like, and needle-like shape populating the same grid and results in a system matrix teeming with different eigenvalues due to the varying metric coefficients. Hence, preconditioning is required to attain fast convergence and the test focuses on the gain by the preconditioner compared to the cost of applying it.

Three cases are investigated here (Fig. 4): $\alpha = 1$, implying a uniform mesh with a maximum aspect ratio of $AR_{max} = 1$; $\alpha = 1.5$, where the maximum aspect ratio in the x_1–x_2 plane is $AR_{max} \approx 17$; and $\alpha = 2$ with $AR_{max} = 128$. The solution process was stopped when the Euclidean norm of the residual is reduced by a factor of 10^{12}. The computations were repeated 11 times. Only the last 10 repetitions contribute to the average runtime, precluding influences from initialization, e.g. library loading. As for time-dependent simulations with implicit diffusion treatment the size of the time step, and thereby the Helmholtz parameter λ, usually change from time step to time step if the time step size is adjusted according to a stability criterion, the precomputation times are included in the measurements. The hardware configuration was the same as employed in Section 7 above.

Fig. 5 summarizes the results of the test. In all three cases, the iteration count behaves similarly: The number of iterations starts at a low value and increases with the polynomial degree, as to be expected. The slope is the largest for UC, slightly lower for DC, and lowest for BC and BT, which exhibit nearly the same iteration count. For the latter three solvers, the iteration count does not differ substantially for different values of α, only an increase by a factor of about 1.5 is observed between $AR_{max} = 1$ and $AR_{max} = 128$. The unpreconditioned solver is not as robust: A factor of four lies between the iteration count for $\alpha = 1$ and the one for $\alpha = 1.5$ and a twenty-fold increase is found between $\alpha = 1$ and $\alpha = 2$. Hence,

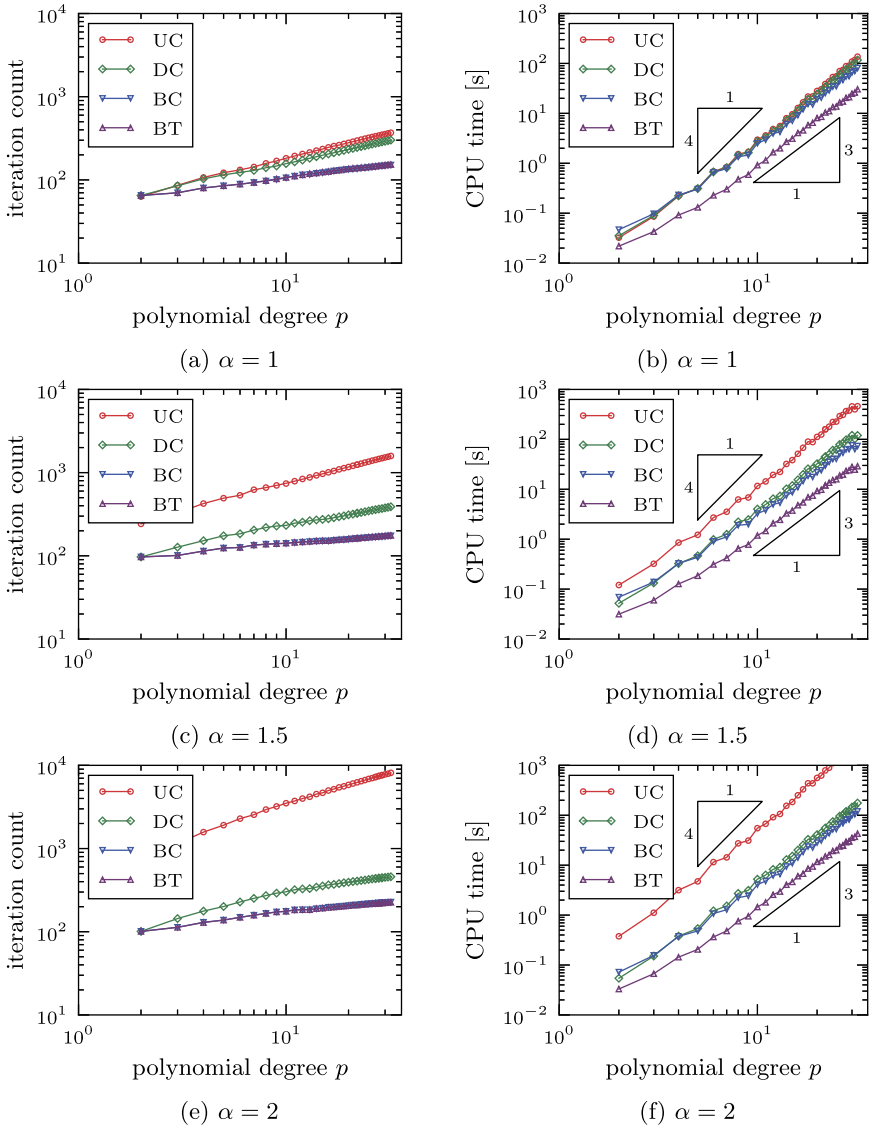

Fig. 5. Iteration count and CPU time for the solution of the linear system using meshes with different expansion factors. (a) and (b): $\alpha = 1$, (c) and (d): $\alpha = 1.5$, (e) and (f): $\alpha = 2$.

when regarding the number of iterations, all preconditioned variants are usable, though DC with some drawbacks compared to BC and BT. But the unpreconditioned one is not usable for non-uniform meshes.

Table 5 lists the runtimes per iteration for the case $\alpha = 2$. As expected, the unpreconditioned solver yields the lowest runtime per iteration for the condensed system, with the diagonal preconditioner slightly increasing the runtime and the block preconditioner taking 50% longer per iteration. However, this advantage is over-compensated by the iteration count of UC: The diagonal preconditioning reduces the runtime of the solver by a factor around 10 for $p = 8$ and approximately by a factor of 17 for $p = 32$. Using the block-preconditioning yields further savings in runtime. Hence, the large effect of the block-preconditioning on the iteration count is mitigated by the runtime spent for preconditioning: The solver DC uses only a quarter more runtime for polynomial degrees $p \leq 16$ than BC and requires less implementation effort.

The solver BC results in a large operator runtime and involves a preconditioner which is too costly. These drawbacks are removed with BT: The operator is faster and the preconditioner is diagonal in the transformed system and, hence, cheap to apply, while generating the same iteration count. Combining both properties leads to a performance gain by a factor of 3 to 4 compared to the diagonally preconditioned case and a factor of 2 to 3 compared to the block-preconditioned version.

While these savings seem insignificant, it has to be kept in mind, that in fact the solvers do not only consist of operator and preconditioner. Many array operations are present in a CG solver and the gather-scatter operation requires runtime as well. The new variant spends most of the time not in applying the operator nor in the preconditioner, but rather multiplying

Table 5
Computation times per degree of freedom and approximated iteration times per degree of freedom as a function of the polynomial degree obtained with the four different solvers using $\alpha = 2$ and $n_e = 8^3$.

p	Iteration time per DOF [ns]				Solution time per DOF [µs]			
	UC	DC	BC	BT	UC	DC	BC	BT
8	52.3	56.2	75.7	25.6	154	15.3	12.6	4.25
12	33.4	36.8	49.2	20.3	133	12.0	9.05	3.73
16	29.1	31.7	41.5	17.7	145	11.6	8.17	3.47
20	27.3	29.5	38.9	16.1	159	11.6	8.02	3.31
24	26.6	28.6	38.8	14.4	177	12.0	8.34	3.09
28	24.9	26.5	36.3	13.1	185	11.7	8.03	2.88
32	23.6	25.2	34.4	12.4	192	11.5	7.81	2.81

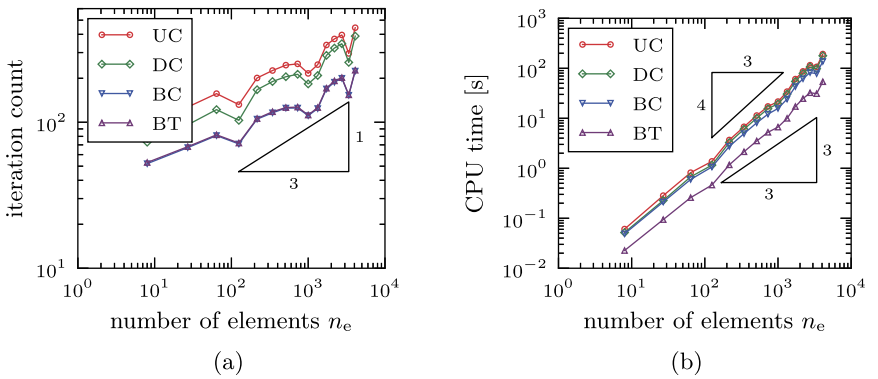

Fig. 6. Iteration count and runtimes, averaged over the last 10 of 11 runs, for $p = 16$ when varying the number of elements.

arrays, etc. For these, no performance gain is possible, so that a hard barrier is present which limits the potential of further factorization.

To investigate the robustness of the solvers against the number of elements n_e, the testcase $\alpha = 1$ was repeated for a constant polynomial degree of $p = 16$ with n_e varying from 2^3 to 16^3. With CG solvers the runtime of a three-dimensional finite element solver generally scales with $n_e^{4/3}$ [20]. Fig. 6 shows the iteration count and the CPU time obtained by averaging over the last 10 of 11 runs of the present SEM solvers. Overall, the number of iterations increases with the number of elements. An exception are drops occurring at $n_e = 5^3, 10^3, 15^3$. These coincide with the wave numbers of the manufactured solution, $5 = k, 10 = 2k, 15 = 3k$ and, hence, are an artifact of the chosen test. The number of iterations exhibits a slightly lower slope than $1/3$. As a result the CPU time scales better than the expected $n_e^{4/3}$. This effect is welcomed, but the reason is probably that the number of elements is still somewhat below the asymptotic regime. Yet the solvers are not robust against an increase in the number of elements, which was to be expected. To achieve this feat, preconditioning with the topology in mind is required, e.g., using low-order finite elements [21,8] or multigrid [22]. This, however, is beyond the scope of the current paper.

9. Conclusions

This paper proposes a new technique for evaluating the condensed Helmholtz operator when using cuboidal elements. It is based on a tensor-product factorization, with a suitably chosen transformation optimizing the multiplication count. The resulting variant of the discrete operator not only scales linearly with the number of degrees of freedom, which was a major goal of this paper, but also reduces the multiplication count to a quarter of that required by an earlier method [9]. This allows the new technique to outpace even variants utilizing highly optimized libraries for matrix–matrix multiplications. An example was provided with the MMC variant that uses DGEMM. Not only does the new method yield a speedup for all polynomial degrees over MMC, by a factor of 20 for $p = 32$ for example. It also achieves a speedup of 2 over previous tensor-product implementations in [9].

After comparing the efficiency of the operators, different solvers based on the two fastest evaluation techniques were investigated using preconditioning with linear scaling in the operation count. Block-Jacobi type preconditioners provided a lower iteration count than diagonal preconditioning, which is nearly independent of the aspect ratio of the elements. For instance an increase of the maximum aspect ratio from $AR_{max} = 1$ to $AR_{max} = 128$ adds just 50% to the iteration count. Yet in the original condensed system, the block preconditioning is far more expensive than in the transformed system. This makes the new solver 2 to 3 times faster than the previous variants. Moreover, the scaling of the operators leads to a linear

scaling of the runtime with respect to the number of degrees of freedom. This was achieved with standard programming language, compiler, and hardware.

While the proposed solver scales very well with respect to the polynomial degree, the performance degrades with the number of elements. This issue was mostly disregarded here, except with some timing measurements. It can be removed by a multigrid approach [22], which is, however, beyond the scope of this paper. Actually, in the transformed system efficient preconditioning can be diagonal which is well suited for operator-based multigrid variants such as cascadic multigrid [23] or multigrid CG methods [24]. Future work will focus on combining the operator factorization laid out in this paper with multigrid techniques to attain a constant iteration count and, hence, to construct a solver scaling linearly in all parameters.

Acknowledgements

This work is supported in part by the German Research Foundation (DFG) within the Cluster of Excellence 'Center for Advancing Electronics Dresden' (cfaed).

References

[1] D. Gottlieb, S.A. Orszag, Numerical Analysis of Spectral Methods: Theory and Applications, CBMS-NSF Regional Conference Series in Applied Mathematics, Society for Industrial and Applied Mathematics, Philadelphia, 1977.
[2] C. Canuto, M. Hussaini, A. Quarteroni, T.A. Zang, Spectral Methods. Fundamentals in Single Domains, Springer, Berlin/Heidelberg, 2006.
[3] A.T. Patera, A spectral element method for fluid dynamics: laminar flow in a channel expansion, J. Comput. Phys. 54 (3) (1984) 468–488, http://dx.doi.org/10.1016/0021-9991(84)90128-1.
[4] W. Couzy, M. Deville, A fast Schur complement method for the spectral element discretization of the incompressible Navier–Stokes equations, J. Comput. Phys. 116 (1) (1995) 135–142, http://dx.doi.org/10.1006/jcph.1995.1011.
[5] S. Yakovlev, D. Moxey, R.M. Kirby, S.J. Sherwin, To CG or to HDG: a comparative study in 3d, J. Sci. Comput. 67 (1) (2015) 192–220, http://dx.doi.org/10.1007/s10915-015-0076-6.
[6] Y.-Y. Kwan, J. Shen, An efficient direct parallel spectral-element solver for separable elliptic problems, J. Comput. Phys. 225 (2) (2007) 1721–1735, http://dx.doi.org/10.1016/j.jcp.2007.02.013.
[7] L. Haupt, J. Stiller, W.E. Nagel, A fast spectral element solver combining static condensation and multigrid techniques, J. Comput. Phys. 255 (2013) 384–395, http://dx.doi.org/10.1016/j.jcp.2013.07.035.
[8] R. Hartmann, M. Lukáčová-Medvid'ová, F. Prill, Efficient preconditioning for the discontinuous Galerkin finite element method by low-order elements, Appl. Numer. Math. 59 (8) (2009) 1737–1753, http://dx.doi.org/10.1016/j.apnum.2009.01.002.
[9] I. Huismann, J. Stiller, J. Fröhlich, Fast static condensation for the Helmholtz equation in a spectral-element discretization, in: Parallel Processing and Applied Mathematics, Springer International Publishing, Cham, 2016, pp. 371–380.
[10] I. Huismann, L. Haupt, J. Stiller, J. Fröhlich, Sum factorization of the static condensed Helmholtz equation in a three-dimensional spectral element discretization, PAMM 14 (1) (2014) 969–970, http://dx.doi.org/10.1002/pamm.201410465.
[11] A.D. Beck, T. Bolemann, D. Flad, H. Frank, G.J. Gassner, F. Hindenlang, C.-D. Munz, High-order discontinuous Galerkin spectral element methods for transitional and turbulent flow simulations, Int. J. Numer. Methods Fluids 76 (8) (2014) 522–548, http://dx.doi.org/10.1002/fld.3943.
[12] J.-E.W. Lombard, D. Moxey, S.J. Sherwin, J.F.A. Hoessler, S. Dhandapani, M.J. Taylor, Implicit large-eddy simulation of a wingtip vortex, AIAA J. 54 (2) (2016) 506–518, http://dx.doi.org/10.2514/1.j054181.
[13] R.E. Lynch, J.R. Rice, D.H. Thomas, Direct solution of partial difference equations by tensor product methods, Numer. Math. 6 (1) (1964) 185–199, http://dx.doi.org/10.1007/bf01386067.
[14] M.O. Deville, P.F. Fischer, E.H. Mund, High-Order Methods for Incompressible Fluid Flow, Cambridge University Press, Cambridge, 2002.
[15] G.E. Karniadakis, S.J. Sherwin, Spectral/hp Element Methods for Computational Fluid Dynamics, 2nd edition, Oxford University Press, Oxford, 2005.
[16] C. Hirsch, Numerical Computation of Internal & External Flows: Fundamentals of Numerical Discretization, John Wiley & Sons, Inc., New York, 1988.
[17] J.J. Dongarra, J.D. Croz, S. Hammarling, I.S. Duff, A set of level 3 basic linear algebra subprograms, ACM Trans. Math. Softw. 16 (1) (1990) 1–17, http://dx.doi.org/10.1145/77626.79170.
[18] E. Merzari, W. Pointer, P. Fischer, Numerical simulation and proper orthogonal decomposition of the flow in a counter-flow T-junction, J. Fluids Eng. 135 (9) (2013) 091304, http://dx.doi.org/10.1115/1.4024059.
[19] M.R. Hestenes, E. Stiefel, Methods of conjugate gradients for solving linear systems, J. Res. Natl. Bur. Stand. 49 (6) (1952) 409–436, http://dx.doi.org/10.6028/jres.049.044.
[20] J.R. Shewchuk, An Introduction to the Conjugate Gradient Method Without the Agonizing Pain, Tech. Rep., Pittsburgh, 1994.
[21] M. Manna, A. Vacca, M.O. Deville, Preconditioned spectral multi-domain discretization of the incompressible Navier–Stokes equations, J. Comput. Phys. 201 (1) (2004) 204–223, http://dx.doi.org/10.1016/j.jcp.2004.05.011.
[22] U. Trottenberg, C. Oosterlee, A. Schüller, Multigrid, Academic Press, Cambridge, 2000.
[23] F.A. Bornemann, P. Deuflhard, The cascadic multigrid method for elliptic problems, Numer. Math. 75 (2) (1996) 135–152, http://dx.doi.org/10.1007/s002110050234.
[24] C. Pflaum, A multigrid conjugate gradient method, Appl. Numer. Math. 58 (12) (2008) 1803–1817, http://dx.doi.org/10.1016/j.apnum.2007.11.020.