p-multigrid methods and their comparison to h-multigrid methods within Isogeometric Analysis

Tielen, R.; Möller, M.; Göddeke, D.; Vuik, C.

DOI
10.1016/j.cma.2020.113347

Publication date
2020

Document Version
Final published version

Published in
Computer Methods in Applied Mechanics and Engineering

Citation (APA)
Tielen, R., Möller, M., Göddeke, D., & Vuik, C. (2020). p-multigrid methods and their comparison to h-multigrid methods within Isogeometric Analysis. *Computer Methods in Applied Mechanics and Engineering*, *372*, 1-27. Article 113347. https://doi.org/10.1016/j.cma.2020.113347

Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 372 (2020) 113347
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

p-multigrid methods and their comparison to h-multigrid methods within Isogeometric Analysis

R. Tielen[a,*], M. Möller[a], D. Göddeke[b,c], C. Vuik[a]

[a] Delft Institute of Applied Mathematics, Delft University of Technology, Van Mourik Broekmanweg 6, 2628XE, Delft, The Netherlands
[b] Institute for Applied Analysis and Numerical Simulation, University of Stuttgart, Allmandring 5b, 70569, Stuttgart, Germany
[c] Stuttgart Center for Simulation Science, University of Stuttgart, Pfaffenwaldring 5c, 70569, Stuttgart, Germany

Received 25 September 2019; received in revised form 30 July 2020; accepted 31 July 2020
Available online xxxx

Abstract

Over the years, Isogeometric Analysis has shown to be a successful alternative to the Finite Element Method (FEM). However, solving the resulting linear systems of equations efficiently remains a challenging task. In this paper, we consider a p-multigrid method, in which coarsening is applied in the spline degree p instead of the mesh width h, and compare it to h-multigrid methods. Since the use of classical smoothers (e.g. Gauss–Seidel) results in a p-multigrid/h-multigrid method with deteriorating performance for higher values of p, the use of an ILUT smoother is investigated as well. Numerical results and a spectral analysis indicate that the use of this smoother exhibits convergence rates essentially independent of h and p for both p-multigrid and h-multigrid methods. In particular, we compare both coarsening strategies (e.g. coarsening in h or p) adopting both smoothers for a variety of two and three dimensional benchmarks. Furthermore, the ILUT smoother is compared to a state-of-the-art smoother (Hofreither and Takacs 2017) using both coarsening strategies. Finally, the proposed p-multigrid method is used to solve linear systems resulting from THB-spline discretizations.

Keywords: Isogeometric analysis; Multigrid methods; p-multigrid; ILUT smoother

1. Introduction

Isogeometric Analysis (IgA) [1] has become widely accepted over the years as an alternative to the Finite Element Method (FEM). The use of B-spline basis functions or Non-Uniform Rational B-splines (NURBS) allows for a highly accurate representation of complex geometries and establishes the link between computer-aided design (CAD) and computer-aided engineering (CAE) tools. Furthermore, the C^{p-1} continuity of the basis functions offers a higher accuracy per degree of freedom compared to standard FEM [2].

IgA has been applied with success in a wide range of engineering fields, such as structural mechanics [3], fluid dynamics [4] and shape optimization [5]. Solving the resulting linear systems efficiently is, however, still a challenging task. The condition numbers of the mass and stiffness matrices increase exponentially with the spline degree p, making the use of (standard) iterative solvers inefficient. On the other hand, the use of (sparse) direct

* Corresponding author.
E-mail address: r.p.w.m.tielen@tudelft.nl (R. Tielen).

https://doi.org/10.1016/j.cma.2020.113347

solvers is not straightforward due to the increasing stencil of the basis functions and increasing bandwidth of matrices for higher values of p. Furthermore, direct solvers may not be practical for large problem sizes due to memory constraints, which is a common problem in high-order methods in general.

Recently, various solution techniques have been developed for discretizations arising in Isogeometric Analysis. For example, preconditioners have been developed based on fast solvers for the Sylvester equation [6] and overlapping Schwarz methods [7].

As an alternative, geometric multigrid methods have been investigated, as they are considered among the most efficient solvers in Finite Element Methods for elliptic problems. However, the use of standard smoothers like (damped) Jacobi or Gauss–Seidel leads to convergence rates which deteriorate for increasing values of p [8]. It has been noted in [9] that very small eigenvalues associated with high-frequency eigenvectors cause this behaviour. This has lead to the development of non-classical smoothers, such as smoothers based on mass smoothing [10–12] or overlapping multiplicative Schwarz methods [13], showing convergence rates independent of both h and p.

p-Multigrid methods can be adopted as an alternative solution strategy. In contrast to h-multigrid methods, a hierarchy is constructed where each level represents a different approximation order. Throughout this paper, the coarse grid correction is obtained at level $p = 1$. Here, B-spline functions coincide with linear Lagrange basis functions, thereby enabling the use of well known solution techniques for standard Lagrangian Finite Elements. Furthermore, the stencil of the basis functions and bandwidth of the matrix is significantly smaller at level $p = 1$, reducing both assembly and factorization costs of the multigrid method.

p-Multigrid methods have mostly been used for solving linear systems arising within the Discontinuous Galerkin method [14–17], where a hierarchy was constructed until level $p = 0$. However, some research has been performed for continuous Galerkin methods [18] as well, where the coarse grid correction was obtained at level $p = 1$.

Recently, the authors applied a p-multigrid method, using a Gauss–Seidel smoother, in the context of IgA [19]. As with h-multigrid methods, a dependence of the convergence rate on p was reported. In this paper, a p-multigrid method is presented that makes use of an Incomplete LU factorization based on a dual Threshold strategy (ILUT) [20] for smoothing. The spectral properties of the resulting p-multigrid method are analysed adopting both smoothers. Numerical results are presented for Poisson's equation on a quarter annulus, an L-shaped (multipatch) geometry and the unit cube. Furthermore, the convection–diffusion–reaction (CDR) equation is considered on the unit square. The use of ILUT as a smoother improves the performance of the p-multigrid method significantly and leads to convergence rates which are essentially independent of h and p.

Compared to standard h-multigrid methods, both the coarsening strategy and smoother are adjusted in the proposed p-multigrid method. Therefore, a comparison study is performed in terms of convergence rates and CPU times between p-multigrid and h-multigrid methods using both smoothers. Furthermore, the p-multigrid method with ILUT as a smoother is compared to an h-multigrid method adopting a smoother based on stable splittings of spline spaces [11]. Finally, to show the versatility of the proposed p-multigrid method, it is applied to solve linear systems of equations resulting from THB-spline discretizations [21].

This paper is organized as follows. In Section 2 the model problem, the basics of IgA and the spatial discretization are considered. Section 3 presents the p-multigrid method in detail, together with the proposed ILUT smoother. A spectral analysis is performed with both smoothers and coarsening strategies and discussed in Section 4. In Section 5, numerical results for the considered benchmarks are presented. Finally, conclusions are drawn in Section 6.

2. Model problem and IgA discretization

To assess the quality of the p-multigrid method, the convection–diffusion–reaction (CDR) equation is considered as a model problem:

$$-\nabla \cdot (\mathbf{D}\nabla u) + \mathbf{v} \cdot \nabla u + Ru = f, \quad \text{on } \Omega, \tag{1}$$

where $\mathbf{D}$ denotes the diffusion tensor, $\mathbf{v}$ a divergence-free velocity field and R a source term. Here, $\Omega \subset \mathbb{R}^2$ is a connected, Lipschitz domain, $f \in L^2(\Omega)$ and $u = 0$ on the boundary $\partial\Omega$. Let $\mathcal{V} = H_0^1(\Omega)$ denote the space of functions in the Sobolev space $H^1(\Omega)$ that vanish on $\partial\Omega$. The variational form of (1) is then obtained by multiplication with an arbitrary test function $v \in \mathcal{V}$ and application of integration by parts: Find $u \in \mathcal{V}$ such that

$$a(u, v) = (f, v) \qquad \forall v \in \mathcal{V}, \tag{2}$$

where

$$a(u, v) = \int_{\Omega} (\mathbf{D}\nabla u) \cdot \nabla v + (\mathbf{v} \cdot \nabla u)v + Ruv \, \mathrm{d}\Omega \tag{3}$$

and

$$(f, v) = \int_{\Omega} fv \, \mathrm{d}\Omega. \tag{4}$$

The physical domain Ω is then parameterized by a geometry function

$$\mathbf{F} : \Omega_0 \to \Omega, \qquad \mathbf{F}(\boldsymbol{\xi}) = \mathbf{x}. \tag{5}$$

The geometry function $\mathbf{F}$ describes an invertible mapping connecting the parameter domain $\Omega_0 = (0, 1)^2$ with the physical domain Ω. In case Ω cannot be described by a single geometry function, the physical domain is divided into a collection of K non-overlapping subdomains $\Omega^{(k)}$ such that

$$\overline{\Omega} = \bigcup_{k=1}^{K} \overline{\Omega}^{(k)}. \tag{6}$$

A family of geometry functions $\mathbf{F}^{(k)}$ is then defined to parameterize each subdomain $\Omega^{(k)}$ separately:

$$\mathbf{F}^{(k)} : \Omega_0 \to \Omega^{(k)}, \qquad \mathbf{F}^{(k)}(\boldsymbol{\xi}) = \mathbf{x}. \tag{7}$$

In this case, we refer to Ω as a multipatch geometry consisting of K patches.

B-spline basis functions

Throughout this paper, the tensor product of univariate B-spline basis functions of order p is used for spatial discretization, unless stated otherwise. Univariate B-spline basis functions are defined on the parameter domain Ω_0 and are uniquely determined by their underlying knot vector

$$\Xi = \{\xi_1, \xi_2, \ldots, \xi_{N+p}, \xi_{N+p+1}\}, \tag{8}$$

consisting of a sequence of non-decreasing knots $\xi_i \in \Omega_0$. Here, N denotes the number of univariate basis functions of order p defined by this knot vector.

B-spline basis functions are defined recursively by the Cox–de Boor formula [22]. The resulting B-spline basis functions $\phi_{i,p}$ are non-zero on the interval $[\xi_i, \xi_{i+p+1})$, implying a compact support that increases with p. Furthermore, at every knot ξ_i the basis functions are C^{p-m_i}-continuous, where m_i denotes the multiplicity of knot ξ_i. Finally, the basis functions possess the partition of unity property:

$$\sum_{i=1}^{N} \phi_{i,p}(\xi) = 1 \quad \forall \xi \in [\xi_1, \xi_{n+p+1}]. \tag{9}$$

Throughout this paper, B-spline basis functions are considered based on an open uniform knot vector with knot span size h, implying that the first and last knots are repeated $p + 1$ times. As a consequence, the basis functions considered are C^{p-1} continuous and interpolatory only at the two end points.

For the two-dimensional case, the tensor product of univariate B-spline basis functions $\phi_{i_x,p}(\xi)$ and $\phi_{i_y,q}(\eta)$ of order p and q, respectively, with maximum continuity is adopted for the spatial discretization:

$$\Phi_{\vec{i},\vec{p}}(\boldsymbol{\xi}) := \phi_{i_x,p}(\xi)\phi_{i_y,q}(\eta), \qquad \vec{i} = (i_x, i_y), \;\; \vec{p} = (p, q). \tag{10}$$

Here, $\boldsymbol{\xi} = (\xi, \eta)$ and $\vec{i}$ and $\vec{p}$ are multi-indices, with $i_x = 1, \ldots, n_x$ and $i_y = 1, \ldots, n_y$ denoting the univariate basis functions in the x and y-dimension, respectively. Furthermore, $i = i_x n_x + (i_y - 1)n_y$ assigns a unique index to each pair of univariate basis functions, where $i = 1, \ldots, N_{\text{dof}}$. Here, N_{dof} denotes the number of degrees of freedom, or equivalently, the number of tensor-product basis functions, and depends on both h and p. In this paper, all univariate B-spline basis functions are assumed to be of the same order (i.e. $p = q$). The spline space $\mathcal{V}_{h,p}$ can then be written, using the inverse of the geometry mapping $\mathbf{F}^{-1}$ as pull-back operator, as follows:

$$\mathcal{V}_{h,p} := \text{span}\left\{ \Phi_{\vec{i},\vec{p}} \circ \mathbf{F}^{-1} \right\}_{i=1,\ldots,N_{\text{dof}}}. \tag{11}$$

The Galerkin formulation of (2) becomes: Find $u_{h,p} \in \mathcal{V}_{h,p}$ such that

$$a(u_{h,p}, v_{h,p}) = (f, v_{h,p}) \qquad \forall v_{h,p} \in \mathcal{V}_{h,p}. \tag{12}$$

The discretized problem can be written as a linear system

$$\mathbf{A}_{h,p}\mathbf{u}_{h,p} = \mathbf{f}_{h,p}, \tag{13}$$

where $\mathbf{A}_{h,p}$ denotes the system matrix resulting from this discretization with B-spline basis functions of order p and mesh width h. For a more detailed description of the spatial discretization in Isogeometric Analysis, we refer to [1]. Throughout this paper four benchmarks are considered, to investigate the influence of the geometric factor, the considered coefficients in the CDR-equation, the number of patches and the spatial dimension on the proposed p-multigrid solver.

Benchmark 1. Let Ω be the quarter annulus with an inner and outer radius of 1 and 2, respectively. The coefficients are chosen as follows:

$$\mathbf{D} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \mathbf{v} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \qquad R = 0. \tag{14}$$

Furthermore, homogeneous Dirichlet boundary conditions are applied and the right-hand side is chosen such that the exact solution u is given by:

$$u(x, y) = -(x^2 + y^2 - 1)(x^2 + y^2 - 4)xy^2.$$

Benchmark 2. Here, the unit square is adopted as domain, i.e. $\Omega = [0, 1]^2$, and the coefficients are chosen as follows:

$$\mathbf{D} = \begin{bmatrix} 1.2 & -0.7 \\ -0.4 & 0.9 \end{bmatrix}, \qquad \mathbf{v} = \begin{bmatrix} 0.4 \\ -0.2 \end{bmatrix}, \qquad R = 0.3. \tag{15}$$

Homogeneous Dirichlet boundary conditions are applied and the right-hand side is chosen such that the exact solution u is given by:

$$u(x, y) = \sin(\pi x)\sin(\pi y).$$

Benchmark 3. Let $\Omega = \{[-1, 1] \times [-1, 1]\}\backslash\{[0, 1] \times [0, 1]\}$ be an L-shaped domain. A multipatch geometry is created, by splitting the single patch in each direction uniformly. The coefficients are chosen as follows:

$$\mathbf{D} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \mathbf{v} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \qquad R = 0. \tag{16}$$

The exact solution is given by:

$$u(x, y) = \begin{cases} \sqrt[3]{x^2 + y^2}\sin\left(\frac{2\text{atan2}(y,x) - \pi}{3}\right) & \text{if } y > 0 \\ \sqrt[3]{x^2 + y^2}\sin\left(\frac{2\text{atan2}(y,x) + 3\pi}{3}\right) & \text{if } y < 0, \end{cases}$$

and the right-hand side is chosen accordingly. Inhomogeneous Dirichlet boundary conditions are prescribed for this benchmark.

Benchmark 4. Here, the unit cube is adopted as domain, i.e. $\Omega = [0, 1]^3$, and the coefficients are chosen as follows:

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{v} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \qquad R = 0. \tag{17}$$

Homogeneous Dirichlet boundary conditions are applied and the right-hand side is chosen such that the exact solution u is given by:

$$u(x, y) = \sin(\pi x)\sin(\pi y)\sin(\pi z).$$

3. p-Multigrid method

Multigrid methods [23,24] aim to solve linear systems of equations by defining a hierarchy of discretizations. At each level of the multigrid hierarchy a smoother is applied, whereas on the coarsest level a correction is determined by means of a direct solver. Starting from $\mathcal{V}_{h,1}$, a sequence of spaces $\mathcal{V}_{h,1}, \ldots, \mathcal{V}_{h,p}$ is obtained by applying k-refinement (i.e. increasing the degree and continuity of the basis functions) to solve Eq. (13). Note that, since basis functions with maximal continuity are considered, the spaces are not nested. A single step of the two-grid correction scheme for the p-multigrid method consists of the following steps [19]:

1. Starting from an initial guess $\mathbf{u}_{h,p}^{(0,0)}$, apply a fixed number ν_1 of pre-smoothing steps:
$$\mathbf{u}_{h,p}^{(0,m)} = \mathbf{u}_{h,p}^{(0,m-1)} + \mathcal{S}_{h,p}\left(\mathbf{f}_{h,p} - \mathbf{A}_{h,p}\mathbf{u}_{h,p}^{(0,m-1)}\right), \quad m = 1, \ldots, \nu_1, \tag{18}$$
where $\mathcal{S}_{h,p}$ is a smoothing operator applied to the high-order problem.
2. Determine the residual at level p and project it onto the space $\mathcal{V}_{h,p-1}$ using the restriction operator $\mathcal{I}_p^{p-1}$:
$$\mathbf{r}_{h,p-1} = \mathcal{I}_p^{p-1}\left(\mathbf{f}_{h,p} - \mathbf{A}_{h,p}\mathbf{u}_{h,p}^{(0,\nu_1)}\right). \tag{19}$$
3. Solve the residual equation at level $p-1$ to determine the coarse grid error:
$$\mathbf{A}_{h,p-1}\mathbf{e}_{h,p-1} = \mathbf{r}_{h,p-1}. \tag{20}$$
4. Project the error $\mathbf{e}_{h,p-1}$ onto the space $\mathcal{V}_{h,p}$ using the prolongation operator $\mathcal{I}_{p-1}^{p}$ and update $\mathbf{u}_{h,p}^{(0,\nu_1)}$:
$$\mathbf{u}_{h,p}^{(0,\nu_1)} \coloneqq \mathbf{u}_{h,p}^{(0,\nu_1)} + \mathcal{I}_{p-1}^{p}\left(\mathbf{e}_{h,p-1}\right). \tag{21}$$
5. Apply ν_2 postsmoothing steps of (18) to obtain $\mathbf{u}_{h,p}^{(0,\nu_1+\nu_2)} =: \mathbf{u}_{h,p}^{(1,0)}$.

In the literature, steps (2)-(4) are referred to as 'coarse grid correction'. Recursive application of this scheme on Eq. (20) until level $p = 1$ is reached, results in a V-cycle. In contrast to h-multigrid methods, the coarsest problem in p-multigrid can still be large for small values of h. However, since we restrict to level $p = 1$, the coarse grid problem corresponds to a standard low-order Lagrange FEM discretization of the problem at hand. Therefore, we use a standard h-multigrid method to solve the coarse grid problem in our p-multigrid scheme, which is known to be optimal (in particular h-independent) in this case. As a smoother, Gauss–Seidel is applied within the h-multigrid method, as it is both cheap and effective for low degree problems. Applying a single W-cycle using canonical prolongation, weighted restriction and a single smoothing step turned out to be sufficient and has therefore been adopted throughout this paper as coarse grid solver.

Note that, for the p-multigrid method, the residual can be projected directly to level $p = 1$. It was shown in [25] that the performance of the p-multigrid method is hardly affected, while the set-up costs decrease significantly. In Appendix A, numerical results are presented for the first benchmark confirming this observation. Throughout this paper, a direct projection to level $p = 1$ is adopted for the p-multigrid method, see Fig. 1. Results are compared to an h-multigrid method, which is shown as well in Fig. 1.

Prolongation and restriction

To transfer both coarse grid corrections and residuals between different levels of the multigrid hierarchy, prolongation and restriction operators are defined. The prolongation and restriction operator adopted in this paper are based on an L_2 projection and have been used extensively in the literature [26–28]. In this paper, the coarse grid correction at level $p = 1$ is prolongated directly to level p by projection onto the space $\mathcal{V}_{h,p}$. The prolongation operator $\mathcal{I}_1^p : \mathcal{V}_{h,1} \to \mathcal{V}_{h,p}$ is given by

$$\mathcal{I}_1^p(\mathbf{v}_1) = (\mathbf{M}_p)^{-1}\mathbf{P}_1^p\ \mathbf{v}_1. \tag{22}$$

Here, the mass matrix $\mathbf{M}_p$ and transfer matrix $\mathbf{P}_1^p$ are defined as follows:

$$(\mathbf{M}_p)_{(i,j)} \coloneqq \int_\Omega \Phi_{i,p}\,\Phi_{j,p}\,\mathrm{d}\Omega, \qquad (\mathbf{P}_1^p)_{(i,j)} \coloneqq \int_\Omega \Phi_{i,p}\,\Phi_{j,1}\,\mathrm{d}\Omega. \tag{23}$$

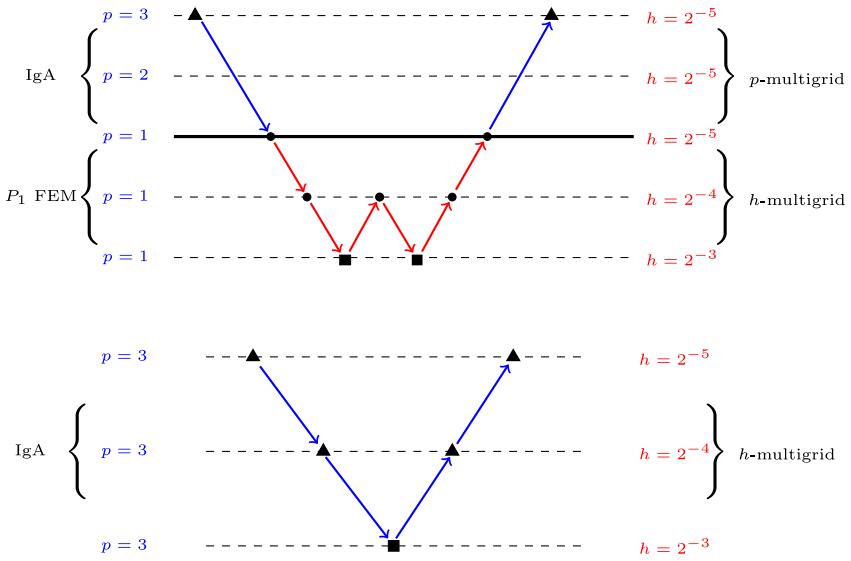

Fig. 1. Illustration of the considered p-multigrid (top) and h-multigrid (bottom) method. At $p = 1$, Gauss–Seidel is always adopted as a smoother (•), whereas at the high order level Gauss–Seidel or ILUT can be applied (▲). At the coarsest level, a direct solver is applied to solve the residual equation (■).

The residuals are restricted from level p to 1 by projection onto the space $\mathcal{V}_{h,1}$. The restriction operator $\mathcal{I}_p^1 : \mathcal{V}_{h,p} \rightarrow \mathcal{V}_{h,1}$ is defined by

$$\mathcal{I}_p^1(\mathbf{v}_p) = (\mathbf{M}_1)^{-1}\mathbf{P}_p^1\ \mathbf{v}_p. \tag{24}$$

To prevent the explicit solution of a linear system of equations for each projection step, the consistent mass matrix $\mathbf{M}$ in both transfer operators is replaced by its lumped counterpart $\mathbf{M}^L$ by applying row-sum lumping:

$$\mathbf{M}_{(i,i)}^L = \sum_{j=1}^{N_{\text{dof}}} \mathbf{M}_{(i,j)}. \tag{25}$$

Numerical experiments, presented in Appendix B, show that lumping the mass matrix hardly influences the convergence behaviour of the resulting p-multigrid method. Neither does it affect the overall accuracy obtained with the p-multigrid method. Alternatively, one could invert the mass matrix efficiently by exploiting the tensor product structure, see e.g. [29].

Note that this choice of prolongation and restriction operators yields a non-symmetric coarse grid correction and, hence, a non-symmetric multigrid solver. As a consequence, the multigrid solver can only be applied as a preconditioner for a Krylov method suited for non-symmetric matrices, like the stabilized Bi-Conjugate Gradient (Bi-CGSTAB) method.

Remark 1. Choosing the prolongation and restriction operator transpose to each other would restore the symmetry of the multigrid method. However, numerical experiments, not presented in this paper, show that this leads to a less robust p-multigrid method. Therefore, the prolongation and restriction operator are adopted as defined in Eqs. (22) and (24), respectively.

Smoother

Within multigrid methods, a basic iterative method is typically used as a smoother. However, in IgA the performance of classical smoothers such as (damped) Jacobi and Gauss–Seidel decreases significantly for higher values of p. Therefore, an Incomplete LU factorization is adopted with a dual Threshold strategy (ILUT) [20] to

approximate the operator $\mathbf{A}_{h,p}$:

$$\mathbf{A}_{h,p} \approx \mathbf{L}_{h,p}\mathbf{U}_{h,p}. \tag{26}$$

The ILUT factorization is determined completely by a tolerance τ and fillfactor m. Two dropping rules are applied during factorization:

1. All elements smaller (in absolute value) than the dropping tolerance are dropped. The dropping tolerance is obtained by multiplying the tolerance τ with the average magnitude of all elements in the current row.
2. Apart from the diagonal element, only the M largest elements are kept in each row. Here, M is determined by multiplying the fillfactor m with the average number of non-zeros in each row of the original operator $\mathbf{A}_{h,p}$.

The ILUT factorization considered in this paper is closely related to an ILU(0) factorization. This factorization has been applied in the context of IgA as a preconditioner, showing good convergence behaviour [30].

An efficient implementation of ILUT is available in the Eigen library [31] based on [32]. Once the factorization is obtained, a single smoothing step is applied as follows:

$$\mathbf{e}_{h,p}^{(n)} = (\mathbf{L}_{h,p}\mathbf{U}_{h,p})^{-1}(\mathbf{f}_{h,p} - \mathbf{A}_{h,p}\mathbf{u}_{h,p}^{(n)}), \tag{27}$$

$$= \mathbf{U}_{h,p}^{-1}\mathbf{L}_{h,p}^{-1}(\mathbf{f}_{h,p} - \mathbf{A}_{h,p}\mathbf{u}_{h,p}^{(n)}), \tag{28}$$

$$\mathbf{u}_{h,p}^{(n+1)} = \mathbf{u}_{h,p}^{(n)} + \mathbf{e}_{h,p}^{(n)}, \tag{29}$$

where the two matrix inversions in Eq. (28) amount to forward and backward substitution. Throughout this paper, the fillfactor $m = 1$ is used (unless stated otherwise) and the dropping tolerance equals $\tau = 10^{-12}$. Hence, the number of non-zero elements of $\mathbf{L}_{h,p} + \mathbf{U}_{h,p}$ is similar to the number of non-zero elements of $\mathbf{A}_{h,p}$. Fig. 2 shows the sparsity pattern of the stiffness matrix $\mathbf{A}_{h,3}$ and $\mathbf{L}_{h,3} + \mathbf{U}_{h,3}$ for the first benchmark and $h = 2^{-5}$. Since an approximate minimum degree (AMD) ordering [33] is applied during the ILUT factorization to reduce the fill-in, sparsity patterns differ significantly. However, the number of non-zero entries is comparable.

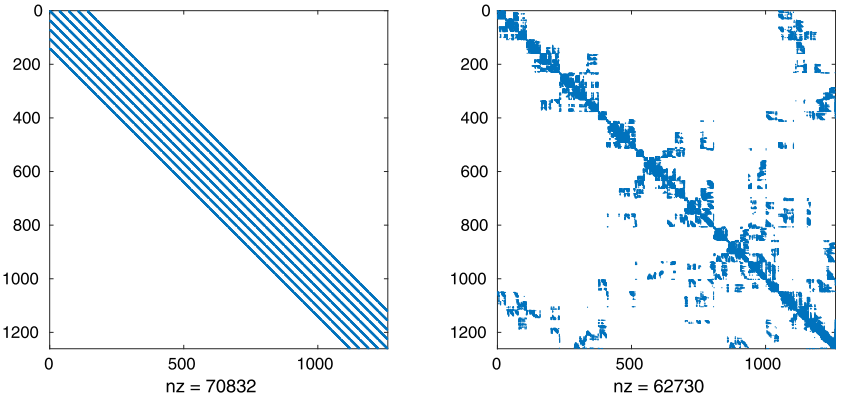

Fig. 2. Sparsity pattern of $\mathbf{A}_{h,3}$ (left) and $\mathbf{L}_{h,3} + \mathbf{U}_{h,3}$ (right) for $h = 2^{-5}$.

Coarse grid operator

At each level of the multigrid hierarchy, the operator $\mathbf{A}_{h,p}$ is needed to apply smoothing or compute the residual. The operators at the coarser levels can be obtained by rediscretizing the bilinear form in (2) with low-order basis functions. Alternatively, a Galerkin projection can be adopted:

$$\mathbf{A}_{h,p-1}^{G} = \mathcal{I}_{p}^{p-1}\ \mathbf{A}_{h,p}\ \mathcal{I}_{p-1}^{p}. \tag{30}$$

However, since the condition number when using the Galerkin projection is significantly higher compared to the condition number of the rediscretized operator, the coarse grid operators in this paper are obtained by using the rediscretization approach.

Computational costs

To investigate the costs of the proposed p-multigrid method, the assembly costs, factorization costs and the costs of a single multigrid cycle are analysed. Assuming a direct projection to level $p = 1$, both $\mathbf{A}_{h,p}$ and $\mathbf{A}_{h,1}$ have to be assembled. Furthermore, the (variationally lumped) mass matrices $\mathbf{M^L}_1$, $\mathbf{M^L}_p$ and transfer matrix $\mathbf{P}_p^1$ have to be assembled.

Assuming an element-based assembly loop with standard Gauss-quadrature, assembling the stiffness matrix or transfer matrix at level p costs $\mathcal{O}(N_{\text{dof}} p^{3d})$ floating point operations (flops). More efficient assembly techniques like weighted quadrature [34], sum factorizations [35] and low tensor rank approximations [36] exist, but will not be explored in this paper. However, since assembly costs might dominate the overall computational costs, assembly costs will be presented separately in Section 5 to illustrate the potential for improving this part of the proposed solution algorithm separately. Assembling the (variationally lumped) mass matrices $\mathbf{M^L}_1$ and $\mathbf{M^L}_p$ costs $\mathcal{O}(N_{\text{dof}})$ flops. At the high order level an ILUT factorization of $\mathbf{A}_{h,p}$ needs to be determined, costing $\mathcal{O}(N_{\text{dof}} p^{2d})$ flops [30] in case $m = 1$ and $\tau = 10^{-12}$. Alternatively to ILUT, Gauss–Seidel can be applied as a smoother, without any set-up costs.

At the high order level both pre- and postsmoothing is applied. Given the ILUT factorization, applying a single smoothing step costs $\mathcal{O}(N_{\text{dof}} p^d)$ flops. Applying Gauss–Seidel as a smoother at level $p = 1$, costs $\mathcal{O}(N_{\text{dof}} 1^d)$ flops [30]. For both prolongation and restriction, a sparse matrix–vector multiplication has to be performed, costing $\mathcal{O}(N_{\text{dof}} p^d)$ flops for each application.

Finally, the residual equation (20) is approximately solved by applying a single W-cycle of an h-multigrid method, which uses Gauss–Seidel as a smoother. Prolongation and restriction operators of the h-multigrid method are based on canonical interpolation and weighted restriction, respectively. Table 1 provides an overview of the computational costs of the proposed p-multigrid method at level k (where k equals 1 or p). Note that, due to the element-based assembly loop, assembly costs dominate the set-up costs. Furthermore, the ILUT factorization leads to significant set-up costs, in particular for higher values of p.

Table 1
Overview of the computational costs with p-multigrid for general values of the approximation order p and dimension d.

Set-up costs				
Level k	Assembly $\mathbf{A}_{h,k}$	Assembly $\mathbf{M^L}_k$	Assembly $\mathbf{P}_k^{k-1}$	ILUT factorization
1	$\mathcal{O}(N_{\text{dof}} 1^{3d})$	$\mathcal{O}(N_{\text{dof}})$		$\mathcal{O}(N_{\text{dof}} 1^{2d})$
p	$\mathcal{O}(N_{\text{dof}} p^{3d})$	$\mathcal{O}(N_{\text{dof}})$	$\mathcal{O}(N_{\text{dof}} p^{3d})$	$\mathcal{O}(N_{\text{dof}} p^{2d})$
Costs multigrid cycle				
Level k	Presmoothing	Restriction	Prolongation	Postsmoothing
1	–	–	$\mathcal{O}(N_{\text{dof}} p^d)$	–
p	$\mathcal{O}(N_{\text{dof}} p^d)$	$\mathcal{O}(N_{\text{dof}} p^d)$	–	$\mathcal{O}(N_{\text{dof}} p^d)$

The memory requirements of the proposed p-multigrid method are strongly related to the number of nonzero entries of each operator. For the stiffness matrix in d dimensions, the number of nonzero entries at level k equals $\mathcal{O}(N_{\text{dof}} k^d)$. Table 2 shows the number of nonzero entries for all operators in the p-multigrid method for each level.

Table 2
Number of nonzero entries with p-multigrid for general values of the approximation order p and dimension d.

Level k	$\mathbf{A}_{h,k}$	$\mathbf{M}_k$	$\mathbf{P}_k^{k-1}$	ILUT factorization
1	$\mathcal{O}(N_{\text{dof}} 1^d)$	$\mathcal{O}(N_{\text{dof}})$	$\mathcal{O}(N_{\text{dof}} 1^d)$	$\mathcal{O}(N_{\text{dof}} 1^d)$
p	$\mathcal{O}(N_{\text{dof}} p^d)$	$\mathcal{O}(N_{\text{dof}})$	$\mathcal{O}(N_{\text{dof}} p^d)$	$\mathcal{O}(N_{\text{dof}} p^d)$

Note that, compared to h-multigrid methods, the p-multigrid method consists of one extra level. Since initially k-refinement is applied, the dimensions of the matrix remain of $\mathcal{O}(N_{\text{dof}})$. However, at level $p = 1$, coarsening in h is applied which leads to a reduction of the number of degrees of freedom with a factor of 2^d from one level to the other, as with h-multigrid. Furthermore, the number of nonzero entries significantly reduces due to the smaller support of the piecewise linear B-spline basis functions. A more detailed comparison between h-multigrid and p-multigrid methods, also in terms of CPU times, can be found in Sections 4 and 5.

4. Spectral analysis

In this section, the performance of the proposed p-multigrid method is analysed and compared with h-multigrid methods in different ways. First, a spectral analysis is performed to investigate the interplay between the coarse grid correction and the smoother. In particular, we compare both smoothers (Gauss–Seidel and ILUT) and coarsening strategies (in h or p). Then, the spectral radius of the iteration matrix is determined to obtain the asymptotic convergence factors of the p-multigrid and h-multigrid methods. Throughout this section, the first two benchmarks presented in Section 2 are considered for the analysis.

Reduction factors

To investigate the effect of a single smoothing step or coarse grid correction, a spectral analysis [37] is carried out for different values of p. For this analysis, we consider $-\Delta u = 0$ with homogeneous Dirichlet boundary conditions and, hence, $u = 0$ as its exact solution. Let us define the error reduction factors as follows:

$$r^{\mathcal{S}}(\mathbf{u}^0_{h,p}) = \frac{\|\mathcal{S}_{h,p}(\mathbf{u}^0_{h,p})\|_2}{\|\mathbf{u}^0_{h,p}\|_2}, \qquad r^{\mathcal{CGC}}(\mathbf{u}^0_{h,p}) = \frac{\|\mathcal{CGC}(\mathbf{u}^0_{h,p})\|_2}{\|\mathbf{u}^0_{h,p}\|_2}, \tag{31}$$

where $\mathcal{S}_{h,p}(\cdot)$ denotes a single smoothing step and $\mathcal{CGC}(\cdot)$ an exact coarse grid correction. We denote a coarse grid correction obtained by coarsening in p and h by $\mathcal{CGC}_p$ and $\mathcal{CGC}_h$, respectively. For $\mathcal{CGC}_p$, a direct projection to $p = 1$ is considered. As an initial guess, the generalized eigenvectors $\mathbf{v}_i$ are chosen which satisfy

$$\mathbf{A}_{h,p}\mathbf{v}_i = \lambda_i \mathbf{M}_{h,p}\mathbf{v}_i, \;\; i = 1, \ldots, N_{\text{dof}}. \tag{32}$$

Here, $\mathbf{M}_{h,p}$ is the consistent mass matrix as defined in (23). The error reduction factors for the first benchmark for both smoothers and coarsening strategies are shown in Fig. 3 for $h = 2^{-5}$ and different values of p. The reduction factors obtained with both smoothers are shown in the left column, while the plots in the right column show the reduction factors for both coarsening strategies. In general, the coarse grid corrections reduce the coefficients of the eigenvector expansion corresponding to the low-frequency modes, while the smoother reduces the coefficients associated with high-frequency modes. However, for increasing values of p, the reduction factors of the Gauss–Seidel smoother increase for the high-frequency modes, implying that the smoother becomes less effective. On the other hand, the use of ILUT as a smoother leads to decreasing reduction factors for all modes when the value of p is increased. The coarse grid correction obtained by coarsening in h (e.g. $\mathcal{CGC}_h$) is more effective compared to a correction obtained by coarsening in p. Note that, for higher values of p, both types of coarse grid correction remain effective in reducing the coefficients of the eigenvector expansion corresponding to the low-frequency modes. The oscillatory behaviour of the reduction rates in Fig. 3 has been observed in the literature for a similar benchmark, see [37], although the exact cause is unknown and requires further investigation. Note that, the reduction rates of the smoother itself are uniformly smaller than 1, implying the smoother is a solver as well.

Fig. 4 shows the error reduction factors obtained for the second benchmark, showing similar, but less oscillatory, behaviour. These results indicate that the use of ILUT as a smoother (with $\nu_1 = \nu_2 = 1$) could significantly improve the convergence properties of the p-multigrid and h-multigrid method compared to the use of Gauss–Seidel as a smoother.

Iteration matrix

For any multigrid method, the asymptotic convergence rate is determined by the spectral radius of the iteration matrix. To obtain this matrix explicitly, consider, again, $-\Delta u = 0$ with homogeneous Dirichlet boundary conditions. By applying a single iteration of the p-multigrid or h-multigrid method using the unit vector $\mathbf{e}^i_{h,p}$ as initial guess, one obtains the ith column of the iteration matrix [38]. Fig. 5 shows the spectra for the first benchmark for $h = 2^{-5}$ and different values of p obtained with both multigrid methods using Gauss–Seidel and ILUT as a smoother. For reference, the unit circle has been added in all plots. The spectral radius of the iteration matrix, defined as the maximum eigenvalue in absolute value, is then given by the eigenvalue that is the furthest away from the origin. Clearly, the spectral radius significantly increases for higher values of p when adopting Gauss–Seidel as a smoother. The use of ILUT as a smoother results in spectra clustered around the origin, implying fast convergence of the resulting p-multigrid or h-multigrid method.

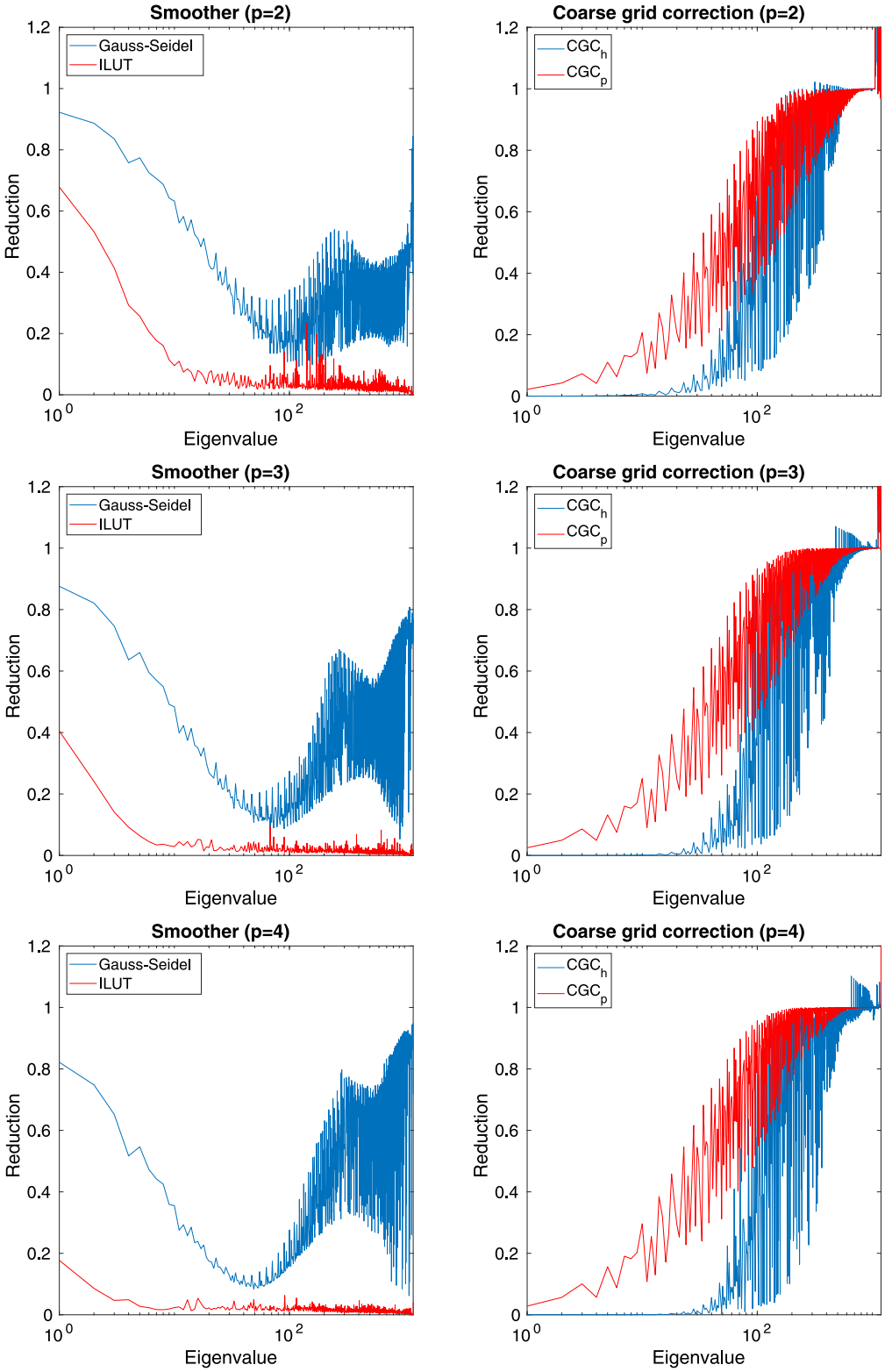

Fig. 3. Error reduction in $(\mathbf{v}_j)$ for the first benchmark with $p = 2, 3, 4$ and $h = 2^{-5}$ obtained with different smoothers (left) and coarsening strategies (right).

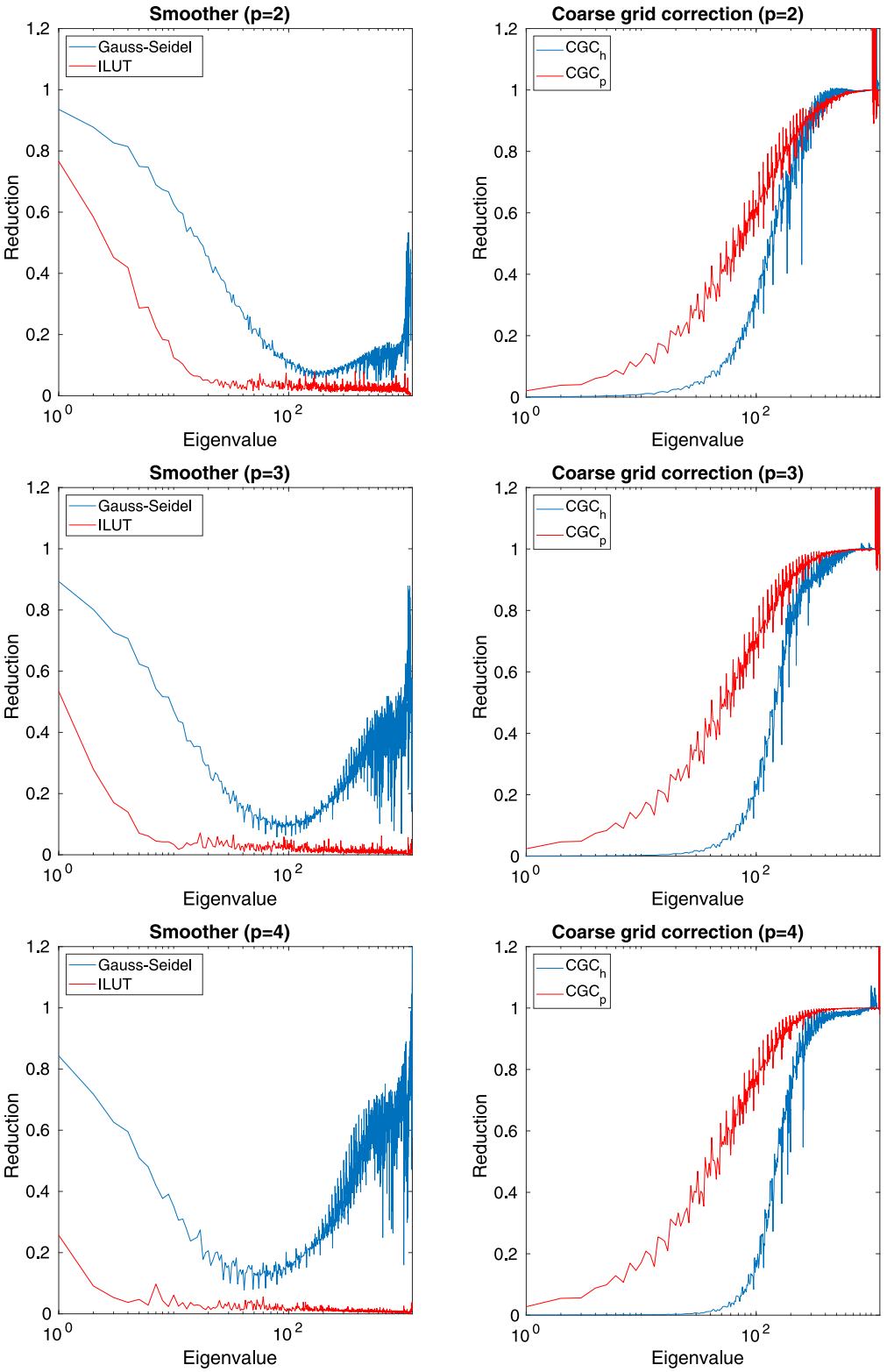

Fig. 4. Error reduction in $(\mathbf{v}_j)$ for the second benchmark with $p = 2, 3, 4$ and $h = 2^{-5}$ obtained with different smoothers (left) and coarsening strategies (right).

The spectra of the iteration matrices for the second benchmark are presented in Fig. 6. Although the eigenvalues are more clustered with Gauss–Seidel compared to the first benchmark, the same behaviour can be observed.

The spectral radii for both benchmarks, where $\nu_1 = \nu_2 = 1$, are presented in Table 3. For Gauss–Seidel, the spectral radius of the iteration matrix is independent of the mesh width h and coarsening strategy, but depends strongly on the approximation order p. The use of ILUT leads to a spectral radius which is significantly lower for all values of h and p. Although ILUT exhibits a small h-dependence, the spectral radius remains low for all values of h and both coarsening strategies. As a consequence, the p-multigrid and h-multigrid method are expected to show essentially both h- and p-independent convergence behaviour when ILUT is adopted as a smoother.

Table 3
Spectral radius for the first and second benchmark using p-multigrid and h-multigrid for different values of the mesh width h and approximation order p.

$p=2$	GS	ILUT	$p=3$	GS	ILUT	$p=4$	GS	ILUT
$h=2^{-4}$	0.635	0.014	$h=2^{-4}$	0.849	0.003	$h=2^{-4}$	0.963	0.003
$h=2^{-5}$	0.631	0.039	$h=2^{-5}$	0.845	0.019	$h=2^{-5}$	0.960	0.029
$h=2^{-6}$	0.647	0.058	$h=2^{-6}$	0.844	0.017	$h=2^{-6}$	0.960	0.023

(a) p-multigrid for the first benchmark

$p=2$	GS	ILUT	$p=3$	GS	ILUT	$p=4$	GS	ILUT
$h=2^{-4}$	0.630	0.012	$h=2^{-4}$	0.848	0.004	$h=2^{-4}$	0.963	0.003
$h=2^{-5}$	0.627	0.039	$h=2^{-5}$	0.845	0.018	$h=2^{-5}$	0.960	0.029
$h=2^{-6}$	0.646	0.059	$h=2^{-6}$	0.844	0.014	$h=2^{-6}$	0.960	0.023

(b) h-multigrid for the first benchmark

$p=2$	GS	ILUT	$p=3$	GS	ILUT	$p=4$	GS	ILUT
$h=2^{-4}$	0.352	0.043	$h=2^{-4}$	0.703	0.002	$h=2^{-4}$	0.916	0.003
$h=2^{-5}$	0.351	0.037	$h=2^{-5}$	0.699	0.011	$h=2^{-5}$	0.913	0.020
$h=2^{-6}$	0.352	0.042	$h=2^{-6}$	0.699	0.017	$h=2^{-6}$	0.914	0.016

(c) p-multigrid for the second benchmark

$p=2$	GS	ILUT	$p=3$	GS	ILUT	$p=4$	GS	ILUT
$h=2^{-4}$	0.367	0.043	$h=2^{-4}$	0.698	0.002	$h=2^{-4}$	0.916	0.003
$h=2^{-5}$	0.367	0.036	$h=2^{-5}$	0.696	0.008	$h=2^{-5}$	0.913	0.020
$h=2^{-6}$	0.359	0.042	$h=2^{-6}$	0.698	0.006	$h=2^{-6}$	0.913	0.016

(d) h-multigrid for the second benchmark

5. Numerical results

In the previous Section, a spectral analysis showed that the use of ILUT as a smoother within a p-multigrid or h-multigrid method significantly improves the asymptotic convergence rate compared to the use of Gauss–Seidel as a smoother. In this Section, p-multigrid and h-multigrid are both applied as a stand-alone solver and as a preconditioner within a Bi-CGSTAB method to verify this analysis. Results in terms of iteration numbers and CPU times are obtained using Gauss–Seidel and ILUT as a smoother. Furthermore, the proposed p-multigrid method is compared to an h-multigrid method using a non-standard smoother. Finally, the p-multigrid method is adopted for discretizations using THB-splines.

For all numerical experiments, the initial guess $\mathbf{u}_{h,p}^{(0)}$ is chosen randomly, where each entry is sampled from a uniform distribution on the interval $[-1, 1]$ using the same seed. Furthermore, we choose $\nu_1 = \nu_2 = 1$ for consistency. Application of multiple smoothing steps, which is in particular common for Gauss–Seidel, decreases the number of iterations until convergence, but does not qualitatively or quantitatively change the p-dependence. Furthermore, since the smoother costs dominate when solving the linear systems, CPU times are only mildly affected. The stopping criterion is based on a relative reduction of the initial residual, where a tolerance of $\epsilon = 10^{-8}$ is adopted. Boundary conditions are imposed by eliminating the degrees of freedom associated to the boundary. For the h-multigrid method, the use of a V-cycle or W-cycle led to the same number of iterations. However, since the computational costs per cycle is lower for V-cycles, they are considered throughout the remainder of this paper for the h-multigrid method.

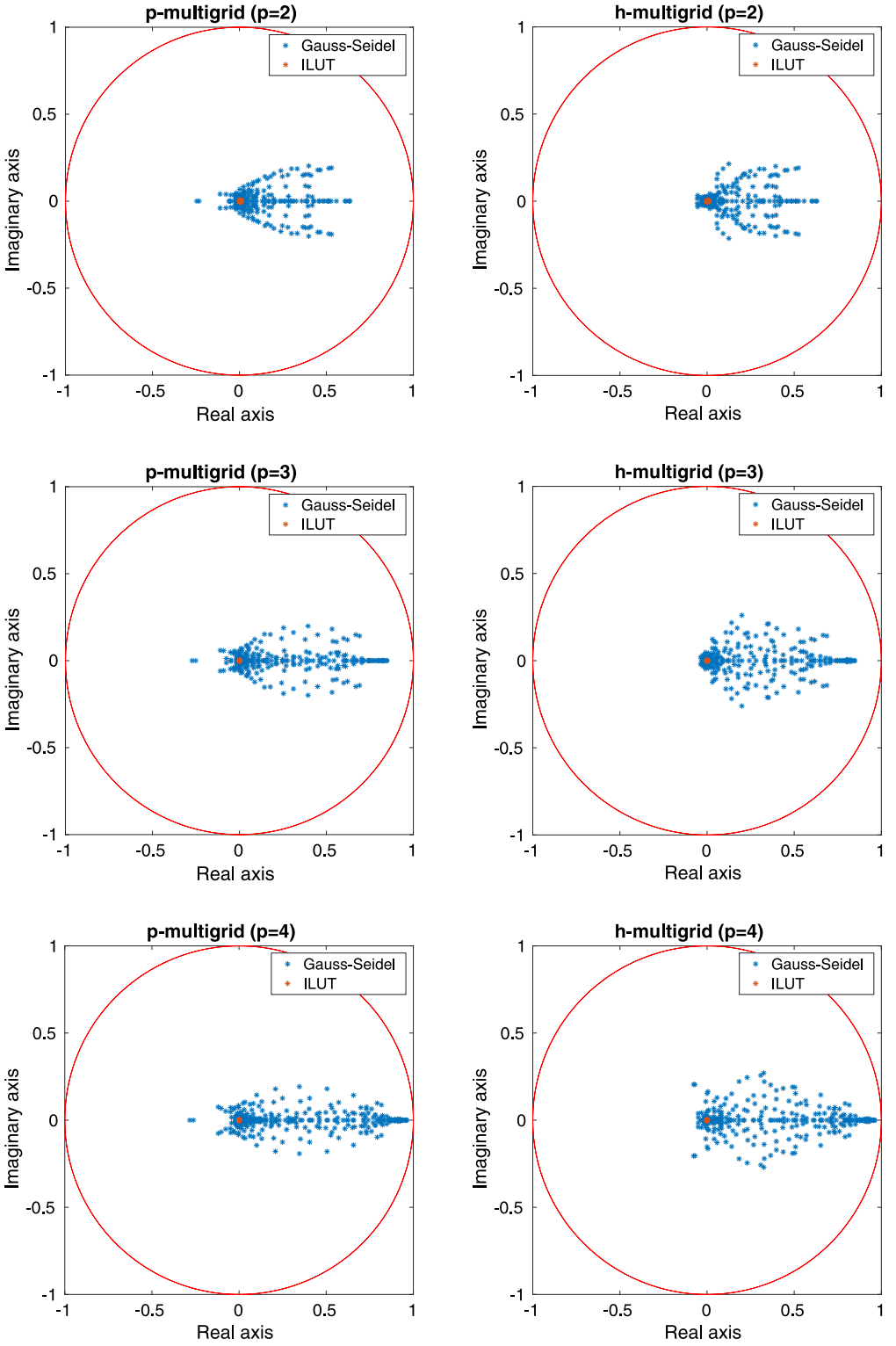

Fig. 5. Spectra of the iteration matrix for the first benchmark obtained with *p*-multigrid (left) and *h*-multigrid (right).

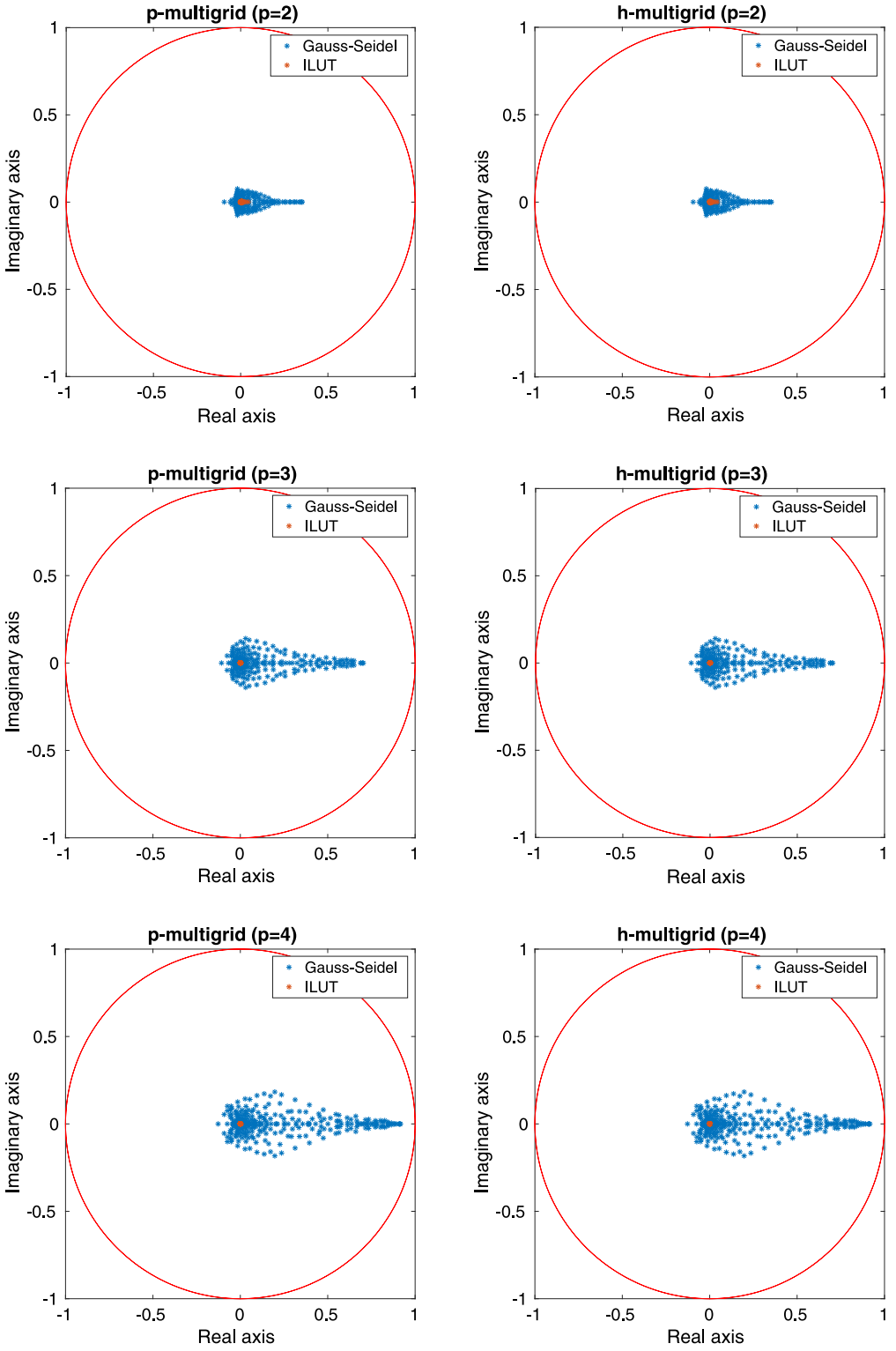

Fig. 6. Spectra of the iteration matrix for the second benchmark obtained with p-multigrid (left) and h-multigrid (right).

p-Multigrid as stand-alone solver

Table 4 shows the number of multigrid cycles needed to achieve convergence using different smoothers for all benchmarks. For the first three benchmarks, the number of multigrid cycles needed with Gauss–Seidel is in general indepen-dent of the mesh width h, but strongly depends on the approximation order p. For some configurations, however, the use of Gauss–Seidel leads to a method that diverges, indicated with $(-)$. The p-multigrid method was said to be diverged in case the norm of the relative residual at the end of a V-cycle was significantly higher than at the end of the previous V-cycle. In general, the use of ILUT as a smoother leads to a p-multigrid which converges for all configurations and exhibits both independence of h and p. Only for the third benchmark, a logarithmic dependence in h is visible for $p = 2$. Furthermore, the number of iterations needed for convergence is significantly lower.

For Poisson's equation on the unit cube (Benchmark 4), the p-dependence when Gauss–Seidel is adopted as smoother is stronger compared to the dependence for the twodimensional benchmarks. Furthermore, the number of iterations slightly decreases for smaller values of the meshwidth h. The number of iterations needed with ILUT as a smoother is effectively independent of the approximation order p. An h-dependence is visible, however, for higher values of p.

h-Multigrid as stand-alone solver

Table 5 shows the number of multigrid cycles needed to achieve convergence using an h-multigrid method. As expected from the spectral analysis, the number of multigrid cycles needed with Gauss–Seidel is in general independent of the mesh width h, but strongly depends on the approximation order p. The use of ILUT as a smoother leads to an h-multigrid which converges for all configurations and exhibits both independence of h and p. Furthermore, the number of iterations needed for convergence is significantly lower. Compared to the use of p-multigrid as a method, the results are very similar. For Benchmark 3, however, the number of iterations needed with h-multigrid using ILUT as a smoother is slightly lower compared to the p-multigrid method.

p-Multigrid as a preconditioner

As an alternative, the p-multigrid method can be applied as a preconditioner within a Bi-CGSTAB method. In both preconditioning phases of each iteration, a single multigrid cycle is applied. Numerical results can be found in Table 6. When applying Gauss–Seidel as a smoother, the number of iterations needed with Bi-CGSTAB is significantly lower compared to the number of p-multigrid cycles and even restores stability for higher values of p (see Table 4). However, a dependence of the iteration numbers on p is still present. When adopting ILUT as a smoother, the number of iterations needed for convergence slightly decreases compared to the number of p-multigrid cycles for all configurations and benchmarks. Furthermore, the number of iterations is independent of both h and p.

h-Multigrid as a preconditioner

Table 7 shows the results when h-multigrid is applied as preconditioner. Note that, since the h-multigrid method is symmetric, a Conjugate Gradient (CG) method can be applied as well. In general, a single iteration performed with a Bi-CGSTAB method is twice as expensive compared to a single CG iteration. Results with a CG method have been added between brackets.

With a Bi-CGSTAB method, results with h-multigrid are very similar to the use of p-multigrid as a preconditioner. Note that, the use of CG as outer Krylov solver, approximately doubles the number of iterations when ILUT is applied as a smoother. For Gauss–Seidel, the use of Bi-CGSTAB yields significant lower iteration numbers compared to the use of CG.

Table 4
Number of multigrid cycles needed to achieve convergence with p-multigrid.

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	GS	ILUT	GS	ILUT	GS	ILUT	GS
$h=2^{-6}$	4	30	3	62	3	176	3	491
$h=2^{-7}$	4	29	3	61	3	172	3	499
$h=2^{-8}$	5	30	3	61	3	163	3	473
$h=2^{-9}$	5	32	3	61	3	163	3	452
(a) Poissons's equation on quarter annulus								
$h=2^{-6}$	5	–	3	–	3	–	4	–
$h=2^{-7}$	5	–	3	–	4	–	4	–
$h=2^{-8}$	5	–	3	–	3	–	4	–
$h=2^{-9}$	5	–	4	–	3	–	4	–
(b) CDR-equation on unit square								
$h=2^{-6}$	6	24	6	53	5	115	5	333
$h=2^{-7}$	7	24	6	53	5	133	5	325
$h=2^{-8}$	8	25	5	54	6	127	6	322
$h=2^{-9}$	9	25	5	55	5	131	5	327
(c) Poissons's equation on L-shaped domain								
$h=2^{-2}$	3	65	3	405	3	3255	5	22 788
$h=2^{-3}$	3	59	3	339	3	2063	3	8128
$h=2^{-4}$	4	57	3	281	3	1652	5	7152
$h=2^{-5}$	4	55	4	273	6	1361	10	6250
(d) Poissons's equation on the unit cube								

Table 5
Number of multigrid cycles needed to achieve convergence with h-multigrid.

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	GS	ILUT	GS	ILUT	GS	ILUT	GS
$h=2^{-6}$	4	30	3	62	3	175	3	492
$h=2^{-7}$	4	29	3	61	3	172	3	499
$h=2^{-8}$	5	30	3	60	3	163	3	473
$h=2^{-9}$	5	32	3	61	3	164	3	452
(a) Poissons's equation on quarter annulus								
$h=2^{-6}$	5	–	3	–	3	–	4	–
$h=2^{-7}$	5	–	3	–	4	–	4	–
$h=2^{-8}$	5	–	3	–	3	–	4	–
$h=2^{-9}$	5	–	3	–	3	–	4	–
(b) CDR-equation on unit square								
$h=2^{-6}$	6	25	3	53	3	113	3	332
$h=2^{-7}$	7	25	3	53	3	133	3	325
$h=2^{-8}$	8	26	3	54	3	127	3	322
$h=2^{-9}$	9	26	3	55	3	131	3	327
(c) Poissons's equation on L-shaped domain								
$h=2^{-2}$	3	65	3	405	3	2269	5	22 785
$h=2^{-3}$	3	59	3	339	3	2065	3	8135
$h=2^{-4}$	3	57	3	281	3	1652	5	7148
$h=2^{-5}$	4	55	3	273	5	1361	10	6248
(d) Poissons's equation on the unit cube								

Table 6
Number of iterations needed to achieve convergence with Bi-CGSTAB, using p-multigrid as preconditioner.

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	GS	ILUT	GS	ILUT	GS	ILUT	GS
$h=2^{-6}$	2	10	2	14	2	27	2	39
$h=2^{-7}$	2	10	2	15	2	25	2	37
$h=2^{-8}$	3	11	2	14	2	25	2	37
$h=2^{-9}$	3	11	2	14	2	26	2	41
(a) Poissons's equation on quarter annulus								
$h=2^{-6}$	2	6	2	11	2	20	2	39
$h=2^{-7}$	2	6	2	11	2	20	2	37
$h=2^{-8}$	2	6	2	11	2	21	2	39
$h=2^{-9}$	2	6	2	10	2	20	2	41
(b) CDR-equation on unit square								
$h=2^{-6}$	3	10	2	13	2	21	2	33
$h=2^{-7}$	3	9	2	12	2	21	2	33
$h=2^{-8}$	3	9	2	12	2	21	2	32
$h=2^{-9}$	4	9	2	13	2	20	2	34
(c) Poissons's equation on L-shaped domain								
$h=2^{-2}$	2	13	2	28	2	77	3	260
$h=2^{-3}$	2	13	2	33	2	65	2	138
$h=2^{-4}$	2	14	2	34	2	72	3	146
$h=2^{-5}$	2	14	2	34	3	69	3	137
(d) Poissons's equation on the unit cube								

Table 7
Number of iterations needed to achieve convergence with Bi-CGSTAB (CG), using h-multigrid as preconditioner.

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	GS	ILUT	GS	ILUT	GS	ILUT	GS
$h=2^{-6}$	2(4)	8(15)	2(3)	15(23)	2(3)	24(42)	2(4)	43(80)
$h=2^{-7}$	2(4)	9(15)	2(3)	15(23)	2(3)	24(42)	2(4)	47(80)
$h=2^{-8}$	3(5)	9(16)	2(3)	14(23)	2(3)	25(41)	2(4)	41(78)
$h=2^{-9}$	3(5)	10(16)	2(3)	14(23)	2(3)	25(41)	2(4)	43(79)
(a) Poissons's equation on quarter annulus								
$h=2^{-6}$	2(4)	6(11)	2(3)	12(18)	2(3)	22(35)	2(4)	36(68)
$h=2^{-7}$	2(4)	7(11)	2(3)	11(18)	2(4)	22(33)	2(4)	40(66)
$h=2^{-8}$	2(4)	7(11)	2(3)	11(18)	2(4)	21(34)	2(4)	39(64)
$h=2^{-9}$	2(4)	6(11)	2(3)	11(18)	2(4)	22(34)	3(4)	40(67)
(b) CDR-equation on unit square								
$h=2^{-6}$	3(5)	9(15)	2(3)	14(22)	2(3)	23(35)	2(4)	39(62)
$h=2^{-7}$	3(5)	10(15)	2(3)	13(22)	2(3)	24(35)	2(3)	37(61)
$h=2^{-8}$	3(6)	10(16)	2(3)	13(22)	2(3)	24(35)	2(4)	35(61)
$h=2^{-9}$	4(7)	10(16)	2(3)	13(22)	2(3)	24(36)	2(4)	40(61)
(c) Poissons's equation on L-shaped domain								
$h=2^{-2}$	2(3)	16(24)	2(3)	37(58)	2(3)	120(158)	3(7)	590(459)
$h=2^{-3}$	2(3)	16(25)	2(3)	48(63)	2(3)	103(168)	2(3)	221(490)
$h=2^{-4}$	2(3)	17(25)	2(3)	43(65)	2(4)	104(188)	3(8)	286(543)
$h=2^{-5}$	2(3)	17(25)	2(3)	44(67)	3(5)	131(197)	3(12)	264(591)
(d) Poissons's equation on the unit cube								

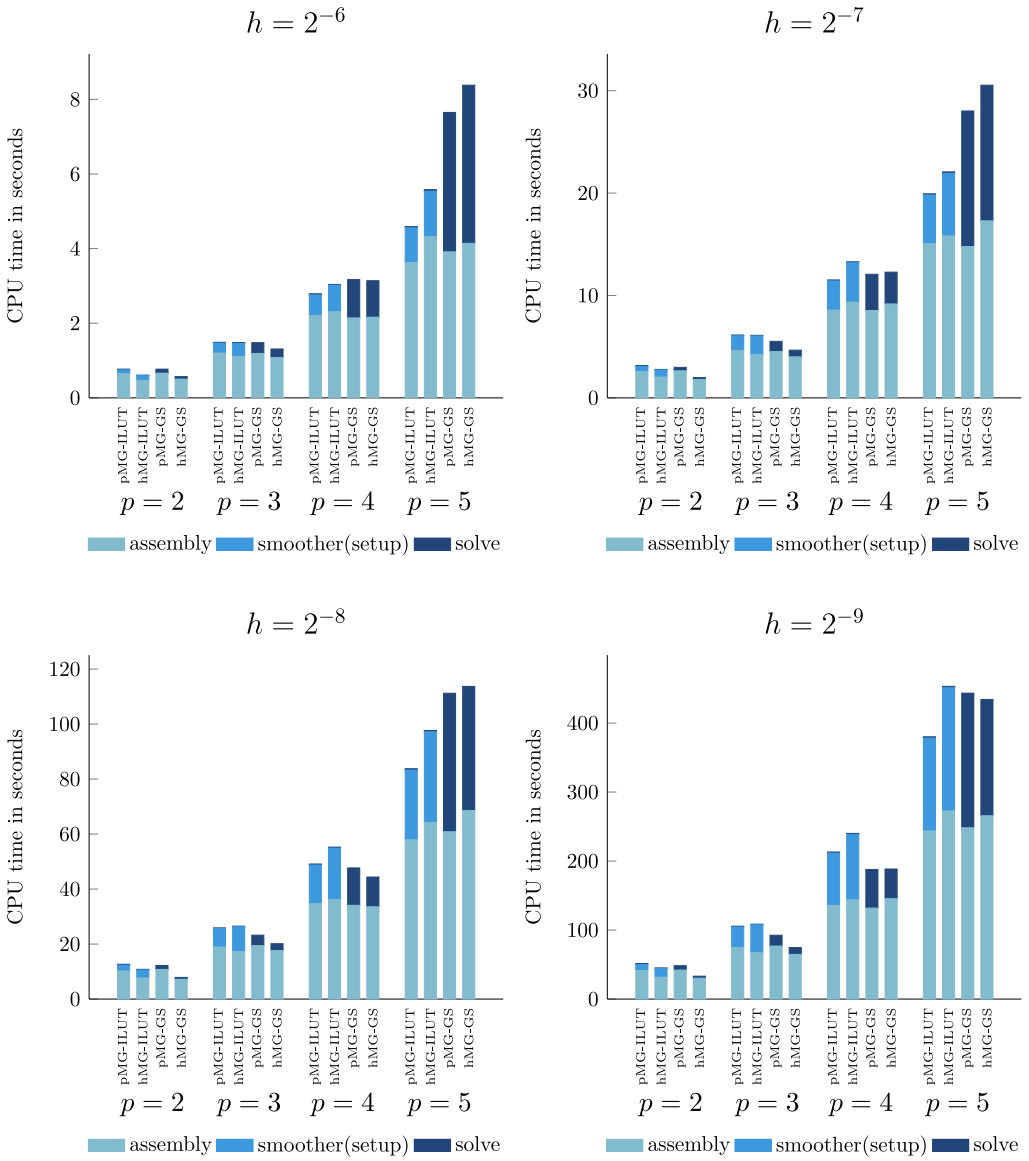

Fig. 7. CPU timings for p-multigrid and h-multigrid adopting ILUT and Gauss–Seidel (GS) as a smoother for different values of h for the first benchmark.

CPU times

Besides iteration numbers, computational times have been determined when adopting p-multigrid and h-multigrid as a stand-alone solver. A serial implementation in the C++ library G+Smo [39] is considered on an Intel(R) Core(TM) i7-8650U CPU (1.90 GHz). Fig. 7 illustrates the CPU times obtained for the p-multigrid and h-multigrid method for the first benchmark. The detailed CPU times can be found in Table C.12 and Table D.13 (see Appendices C and D).

The assembly times denote the CPU time needed to assemble all operators, including the prolongation and restriction operators. Note that, for the p-multigrid method more operators have to be assembled. However, most of the operators in the p-multigrid method are assembled at level $p = 1$, where the number of nonzero entries is significantly lower compared to the matrices resulting from high order discretizations. As a consequence the total assembly costs are lower with p-multigrid compared to h-multigrid for higher values of the approximation order p.

With respect to the setup costs of the smoother, similar observations can be made: For higher values of p, the ILUT factorization costs are significantly higher for the h-multigrid method. The time needed to solve linear systems is slightly lower for the h-multigrid methods, since the costs of a single multigrid cycle are lower compared to the p-multigrid method. When adopting Gauss–Seidel as a smoother, the time needed to solve the linear systems is significantly higher compared to the use of ILUT. However, since the factorizations costs are relatively high, the p-multigrid/h-multigrid methods using ILUT as a smoother are faster for only a limited amount of configurations.

Remark 2. For all numerical experiments, the 'coarse grid' operators of the multigrid hierarchy have been obtained by rediscretizing the bilinear form in Eq. (2). Alternatively, all operators of the h-multigrid hierarchy could be obtained by applying the Galerkin projection. Furthermore, alternative (and more efficient) assembly strategies exist, as mentioned in Section 3. Therefore, the assembly, smoother setup and solving costs are presented separately in this section.

Comparison with an alternative smoother

Throughout this paper, the use of ILUT and Gauss–Seidel as a smoother has been investigated within a p-multigrid and h-multigrid method. However, alternative smoothers have been developed for h-multigrid methods in recent years, for example, a subspace corrected mass smoother (SCMS) [11] based on stable splittings of spline spaces. In this section, we compare the ILUT smoother with this smoother for both coarsening strategies. Note that this smoother has been extended to multipatch domains as well [40]. For this comparison, we consider the CDR-equation on the unit square with:

$$\mathbf{D} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \mathbf{v} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \qquad R = 1. \tag{33}$$

Homogeneous Neumann boundary conditions are applied and the right-hand side is given by:

$$f(x, y) = 2\pi^2 \sin\left(\pi\left(x + \frac{1}{2}\right)\right) \sin\left(\pi\left(y + \frac{1}{2}\right)\right).$$

Table 8 shows the number of iterations needed to reach convergence with the p-multigrid method and the h-multigrid method for the ILUT smoother and the subspace corrected mass smoother. For the ILUT smoother, iteration numbers are independent of h and p for both coarsening strategies. The smoother from [11] shows iteration numbers independent of h and p within a h-multigrid method. A slight p-dependency is visible when this smoother is applied within a p-multigrid method. With the ILUT smoother, the number of iterations needed to reach convergence is significantly lower for all configurations.

Table 8
Number of iterations with ILUT and the smoother from [11] using p-multigrid and h-multigrid.

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	SCMS	ILUT	SCMS	ILUT	SCMS	ILUT	SCMS
$h=2^{-6}$	5	40	5	44	5	47	5	52
$h=2^{-7}$	5	40	5	44	4	48	5	53
$h=2^{-8}$	5	40	4	44	5	48	4	53
$h=2^{-9}$	5	40	4	45	5	48	4	53

(a) Number of iterations with p-multigrid for both smoothers.

	ILUT	SCMS	ILUT	SCMS	ILUT	SCMS	ILUT	SCMS
$h=2^{-6}$	4	48	4	48	4	48	4	48
$h=2^{-7}$	4	49	3	50	4	49	4	49
$h=2^{-8}$	4	49	3	50	5	50	4	49
$h=2^{-9}$	4	49	3	50	5	50	4	50

(b) Number of iterations with h-multigrid for both smoothers.

CPU times for assembly, setting up the smoother and solving the linear system once are presented in Fig. 8 (left). Again, a serial implementation in the C++ library G+Smo [39] is considered on an Intel(R) Core(TM) i7-8650U CPU (1.90 GHz). Detailed CPU times can be found in Tables E.14 and F.15 (see Appendices E and F). The time needed to assemble the operators is comparable for the p-multigrid and h-multigrid method. However, setting up

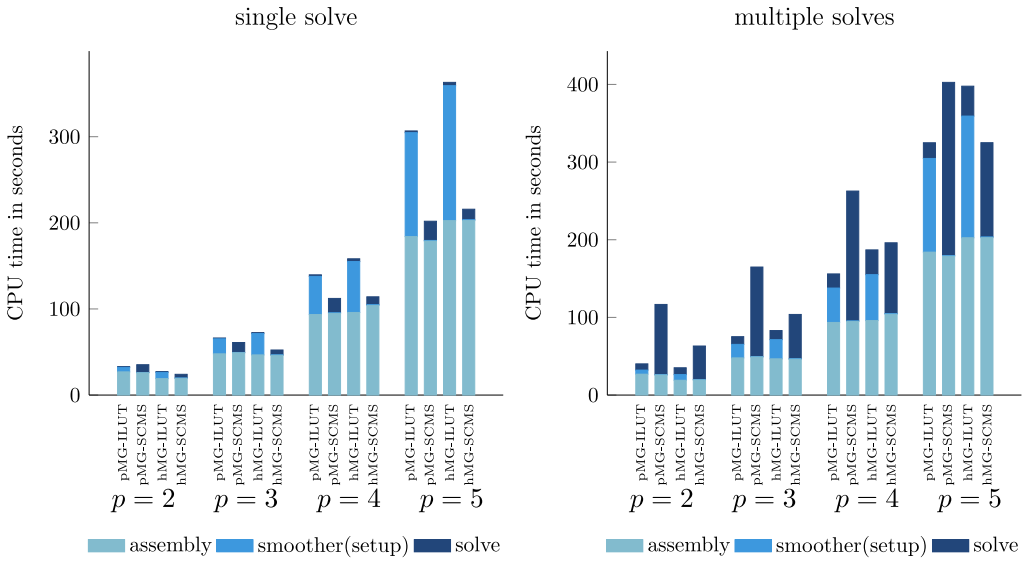

Fig. 8. CPU timings for p-multigrid and h-multigrid adopting ILUT and the smoother from [11], respectively, for a single solve (left) and 10 solves (right).

the ILUT smoother is significantly more expensive compared to the smoother from [11]. On the other hand, the CPU time needed to solve the problem is lower when adopting the ILUT smoother. The total solver costs are lower for all configurations when adopting the subspace corrected mass smoother. However, in case multiple solves are necessary with the same system matrix and different right-hand sides, the ILUT smoother becomes relatively seen more efficient. This is for example the case when 'snapshot' solutions are required to apply Proper Orthogonal Decomposition [41]. Fig. 8 (right) shows that, already when solving the linear system for 10 different right hand sides, the CPU times with ILUT as a smoother within the p-multigrid method are lower for all values of p. For the h-multigrid method, however, the use of the smoother from [11] remains more efficient for high values of p.

Truncated hierarchical B-splines (THB-splines)

Finally, to illustrate the versatility of the proposed p-multigrid method, we consider discretizations obtained with THB-splines [21]. THB-splines are the result of a local refinement strategy, in which a subset of the basis functions on the fine level are truncated. As a result, not only linear independence and non-negativity are preserved (as with HB-splines [42,43]), but also the partition of unity property.

In the literature, the use of multigrid methods for THB-spline discretizations is an ongoing topic of research [44–46]. We consider Poisson's equation on the unit square, where the exact solution is the same as for the second benchmark. Starting from a tensor product B-spline basis with meshwidth h and order p, two and three levels of refinement are added as shown in Fig. 9, leading to a THB-spline basis consisting of, respectively, three and four levels.

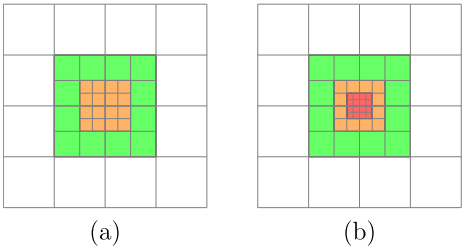

Fig. 9. Two hierarchical mesh adopted for THB-Spline basis with the second (green), third (orange) and fourth (red) refinement levels coloured. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

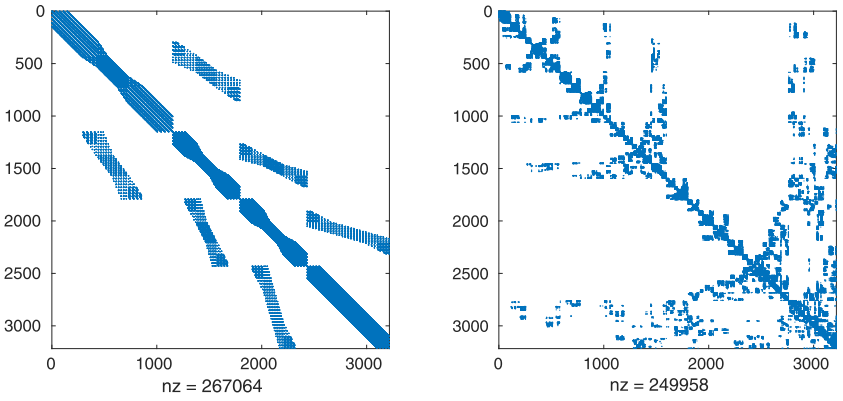

Fig. 10. Sparsity pattern of the stiffness matrix $\mathbf{A}_{h,4}$ (left) and $\mathbf{L}_{h,4} + \mathbf{U}_{h,4}$ (right).

Fig. 10 shows the sparsity pattern of the stiffness matrix and the ILUT factorization for $p = 4$ and $h = 2^{-5}$ for configuration (b) (see Fig. 9). Compared to the (standard) tensor-product B-spline basis the bandwidth of the stiffness matrix significantly increases. Table 9 shows the results obtained with p-multigrid applied as a stand-alone solver. The number of iterations needed with p-multigrid (and ILUT as a smoother) depends only mildly on p. Furthermore, the number of iterations are significantly lower compared to the use of Gauss–Seidel as a smoother.

Table 9
Number of multigrid cycles needed for different THB-spline discretizations.

	$p = 2$		$p = 3$		$p = 4$		$p = 5$	
	ILUT	GS	ILUT	GS	ILUT	GS	ILUT	GS
$h = 2^{-4}$	5	16	6	45	5	178	5	713
$h = 2^{-5}$	5	17	6	40	7	182	**5**	882
$h = 2^{-6}$	5	17	5	41	7	189	**11**	936
(a) THB-spline basis with three levels of refinement								
$h = 2^{-4}$	6	17	8	47	7	177	10	1033
$h = 2^{-5}$	6	16	7	44	8	182	**7**	923
$h = 2^{-6}$	6	17	5	43	6	201	**12**	1009
(b) THB-spline basis with four levels of refinement								

For the configurations denoted in bold, a fillfactor of 2 was adopted, to prevent the p-multigrid from diverging. Fig. 11 illustrates the reason for it in the case $p = 4$ and $h = 2^{-4}$ for configuration (a). A fillfactor of 1 does not reduce the norm of the (generalized) eigenvectors, while a fillfactor of 2 reduces the eigenvectors over the entire spectrum. In general, a higher fillfactor was necessary for only a limited amount of configurations.

For all numerical experiments, smoothing is performed globally at each level of the multigrid hierarchy. In general, local smoothing is often adopted to ensure optimal order of the complexity. Results presented in this Section should be considered as a first step towards the use of p-multigrid methods for THB-spline discretizations. Future research should focus on more efficient applications of p-multigrid solvers for THB-spline discretizations.

6. Conclusions

In this paper, we presented a p-multigrid method that uses ILUT factorization as a smoother and compared this with different smoothers and coarsening strategy (e.g. h-multigrid). In contrast to classical smoothers, (i.e. Gauss–Seidel), the reduction factors of the general eigenvectors associated with high-frequency modes do not increase when adopting ILUT as a smoother for higher values of p. This results in asymptotic convergence factors which are independent of both the mesh width h and approximation order p for both p-multigrid and h-multigrid methods adopting this smoother. Furthermore, we observed that, assuming an exact coarse grid correction, coarsening in h leads to a more effective coarse grid correction compared to a correction obtained by coarsening in p.

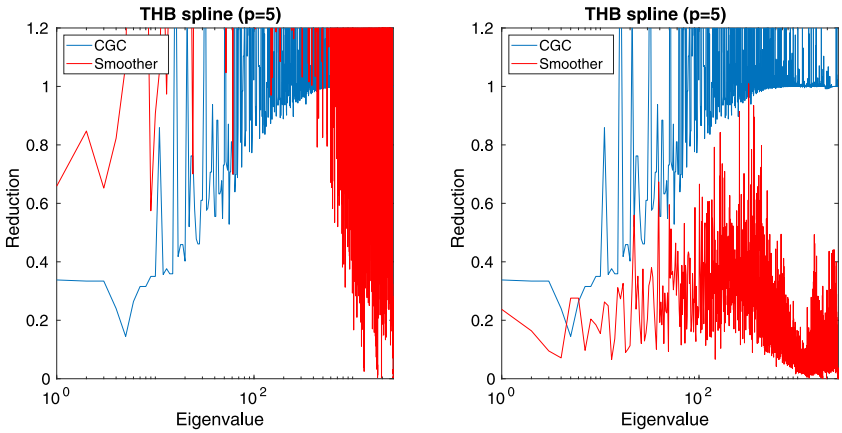

Fig. 11. Reduction factors obtained for fillfactor 1 (left) and 2 (right).

Numerical results, obtained for Poisson's equation on a variety of domains and the CDR equation on the unit square have been presented when using p-multigrid and h-multigrid as stand-alone solver or as a preconditioner within a Bi-CGSTAB or CG method. For all configurations, the number of iterations needed when using ILUT as a smoother are significantly lower compared to the use of Gauss–Seidel, while the number of iterations needed with p-multigrid are very similar to those needed with an h-multigrid method. Hence, the smoother determines to a great extent the resulting convergence rate of the multigrid method. CPU times have been presented for the p-multigrid and h-multigrid method using both smoothers. For low values of p, the use of h-multigrid combined with Gauss–Seidel as a smoother leads to the lowest CPU times. For higher values of p, however, the use of p-multigrid adopting ILUT becomes more efficient, due to the lower assembly and factorizations costs. Note that this is the result of the smaller stencil of the B-spline functions at level $p = 1$ compared to high order B-spline functions.

The p-multigrid method using ILUT as a smoother has been compared as well to an h-multigrid method with a non-standard smoother [11]. Results show that the total solving costs are lower when adopting h-multigrid with this smoother due to the lower setup costs of the smoother. However, solving the linear systems is significantly faster with the considered p-multigrid method. Finally, the p-multigrid method has been applied to solve linear systems of equations arising from THB-spline discretizations. In general, a significantly lower number of iterations was needed with ILUT compared to the use of Gauss–Seidel as a smoother. For a limited number of configurations, a higher fillfactor of 2 (instead of 1) was necessary to achieve convergence.

Future research will focus on the application of p-multigrid methods for higher-order partial differential equations (e.g.. biharmonic equation), where the use of basis functions with high continuity is necessary. Furthermore, local smoothing within the p-multigrid method should be considered to make it more efficient for THB-spline discretizations. Finally, the use of block ILUT as a smoother in case of a multipatch geometry will be investigated.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors would like to thank Prof. Kees Oosterlee from TU Delft for fruitful discussions with respect to p-multigrid methods.

Appendix A. Direct or indirect projection

To investigate the effect of a direct projection to $p = 1$, we consider the first benchmark. The number of multigrid cycles needed to achieve convergence with a direct projection and indirect projection has been determined for different values of h and p. Table A.10 shows the number of iterations needed to achieve convergence with a direct and indirect projection, respectively. For most configurations, the number of iterations is very similar. Only for higher values of p, the indirect project leads to diverging method when Gauss–Seidel is applied as a smoother. With a direct projection, all configurations lead to a converging multigrid method.

Table A.10
Number of multigrid cycles needed to achieve convergence with p-multigrid for the first benchmark with a direct and an indirect projection.

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	GS	ILUT	GS	ILUT	GS	ILUT	GS
$h=2^{-6}$	4	30	3	62	3	176	3	491
$h=2^{-7}$	4	29	3	61	3	172	3	499
$h=2^{-8}$	5	30	3	61	3	163	3	473
$h=2^{-9}$	5	32	3	61	3	163	3	452
(a) p-multigrid with a direct projection								
$h=2^{-6}$	4	30	3	62	3	–	3	–
$h=2^{-7}$	4	29	3	61	3	–	3	–
$h=2^{-8}$	5	30	3	61	3	–	3	–
$h=2^{-9}$	5	32	3	63	3	–	3	–
(b) p-multigrid with an indirect projection								

Appendix B. Consistent vs. lumped projection

In Section 2, the prolongation and restriction operator to transfer residuals and corrections from level p to 1 and vice versa have been defined. Note that, the mass matrix in Eqs. (22) and (24) can be lumped to reduce computational costs. To investigate the effect of lumping the mass matrix within the L_2 projection, the first benchmark is considered.

Table B.11 shows the number of multigrid cycles needed to achieve convergence using the lumped or consistent mass matrix in Eqs. (22) and (24). When ILUT is adopted as a smoother, the number of multigrid cycles needed to reach convergence is identical for all configurations. For Gauss–Seidel, the use of the consistent mass matrix leads to a slightly lower number of iterations. Considering the decrease of computational costs, however, the lumped mass matrix is adopted throughout the entire paper in the prolongation and restriction operator.

Table B.11
Number of multigrid cycles to reach convergence with p-multigrid adopting a lumped or consistent mass matrix in the prolongation and restriction operator.

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	GS	ILUT	GS	ILUT	GS	ILUT	GS
$h=2^{-6}$	4	30	3	62	3	176	3	491
$h=2^{-7}$	4	29	3	61	3	172	3	499
$h=2^{-8}$	5	30	3	61	3	163	3	473
$h=2^{-9}$	5	32	3	63	3	163	3	452
(a) Lumped mass matrix $\mathbf{M}_k^L$								
$h=2^{-6}$	4	29	3	57	3	171	3	475
$h=2^{-7}$	4	29	3	52	3	174	3	524
$h=2^{-8}$	5	29	3	54	3	165	3	446
$h=2^{-9}$	5	31	3	52	3	164	3	441
(b) Consistent mass matrix $\mathbf{M}_k$								

Appendix C. CPU times *p*-multigrid

Table C.12
CPU timings for the first benchmark using p-multigrid.

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	GS	ILUT	GS	ILUT	GS	ILUT	GS
$h=2^{-6}$	0.65	0.66	1.20	1.19	2.21	2.14	3.63	3.91
$h=2^{-7}$	2.58	2.63	4.64	4.52	8.58	8.53	15.06	14.78
$h=2^{-8}$	10.22	10.77	18.93	19.41	34.77	34.07	57.95	60.81
$h=2^{-9}$	41.52	41.86	74.74	76.70	135.79	131.66	243.64	248.11
(a) Assembly costs in seconds								
$h=2^{-6}$	0.10	–	0.27	–	0.55	–	0.93	–
$h=2^{-7}$	0.50	–	1.43	–	2.85	–	4.76	–
$h=2^{-8}$	2.13	–	6.70	–	13.98	–	25.32	–
$h=2^{-9}$	8.66	–	29.78	–	75.89	–	134.57	–
(b) Factorization costs in seconds								
$h=2^{-6}$	0.02	0.11	0.02	0.29	0.03	1.03	0.03	3.74
$h=2^{-7}$	0.07	0.35	0.07	1.00	0.09	3.54	0.11	13.24
$h=2^{-8}$	0.32	1.46	0.26	3.82	0.36	13.62	0.49	50.39
$h=2^{-9}$	1.30	6.48	1.07	15.80	1.46	56.15	1.88	195.28
(c) Solver costs in seconds								
$h=2^{-6}$	0.77	0.77	1.49	1.48	2.79	3.17	4.59	7.65
$h=2^{-7}$	3.15	2.98	6.14	5.52	11.52	12.07	19.93	28.02
$h=2^{-8}$	12.67	12.23	25.89	23.23	49.11	47.69	75.97	111.20
$h=2^{-9}$	51.48	48.34	105.59	92.50	213.14	187.81	380.09	443.39
(d) Total costs in seconds								

Appendix D. CPU times *h*-multigrid

Table D.13
CPU timings for the first benchmark using h-multigrid with different smoothers.

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	GS	ILUT	GS	ILUT	GS	ILUT	GS
$h=2^{-6}$	0.47	0.50	1.11	1.08	2.31	2.16	4.32	4.14
$h=2^{-7}$	2.03	1.80	4.21	4.00	9.35	9.16	15.82	17.28
$h=2^{-8}$	7.68	7.17	17.33	17.64	36.17	33.53	64.25	68.57
$h=2^{-9}$	31.77	30.14	67.10	64.69	143.86	145.38	272.68	265.49
(a) Assembly costs in seconds								
$h=2^{-6}$	0.13	–	0.35	–	0.71	–	1.22	–
$h=2^{-7}$	0.70	–	1.84	–	3.85	–	6.12	–
$h=2^{-8}$	2.93	–	8.98	–	18.74	–	32.94	–
$h=2^{-9}$	12.52	–	40.53	–	94.89	–	178.58	–
(b) Setup costs smoother in seconds								
$h=2^{-6}$	0.01	0.07	0.02	0.23	0.02	0.98	0.04	4.24
$h=2^{-7}$	0.05	0.18	0.06	0.66	0.09	3.12	0.13	13.26
$h=2^{-8}$	0.21	0.70	0.21	2.49	0.34	10.85	0.49	45.14
$h=2^{-9}$	0.86	3.01	0.87	9.93	1.35	43.12	1.94	168.74
(c) Solver costs in seconds								
$h=2^{-6}$	0.61	0.57	1.48	1.31	3.05	3.14	5.59	8.39
$h=2^{-7}$	2.78	1.98	6.11	4.66	13.29	12.28	22.07	30.54
$h=2^{-8}$	10.82	7.87	26.52	20.13	55.25	44.38	97.68	113.71
$h=2^{-9}$	45.15	33.15	108.50	74.62	240.10	188.50	453.20	434.22
(d) Total costs in seconds								

Appendix E. CPU times compared to an alternative smoother

Table E.14
CPU times (in s) for convergence with p-multigrid.

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	SCMS	ILUT	SCMS	ILUT	SCMS	ILUT	SCMS
$h=2^{-6}$	0.42	0.41	0.76	0.81	1.46	1.42	2.70	2.84
$h=2^{-7}$	1.62	1.62	3.04	2.94	6.26	5.91	11.74	11.22
$h=2^{-8}$	6.50	6.65	12.47	12.96	24.19	24.64	46.63	43.60
$h=2^{-9}$	26.91	25.86	47.85	49.10	93.50	94.84	184.05	178.78
(a) Assembly costs in seconds								
$h=2^{-6}$	0.06	0.01	0.19	0.01	0.41	0.01	0.78	0.02
$h=2^{-7}$	0.30	0.02	0.96	0.03	2.20	0.04	4.64	0.05
$h=2^{-8}$	1.25	0.09	4.16	0.10	10.64	0.17	22.53	0.20
$h=2^{-9}$	5.26	0.34	17.44	0.37	44.30	0.66	120.64	0.77
(b) Setup costs smoother in seconds								
$h=2^{-6}$	0.02	0.19	0.02	0.26	0.03	0.31	0.05	0.44
$h=2^{-7}$	0.06	0.62	0.09	0.77	0.09	1.08	0.16	1.46
$h=2^{-8}$	0.23	2.08	0.26	2.76	0.48	4.03	0.54	5.51
$h=2^{-9}$	0.80	9.05	0.99	11.54	1.82	16.72	2.02	22.31
(c) Solver costs in seconds								
$h=2^{-6}$	0.50	0.61	0.97	1.08	1.90	1.74	3.53	3.30
$h=2^{-7}$	1.98	2.26	4.09	3.74	8.55	7.03	16.54	12.73
$h=2^{-8}$	7.98	8.82	16.89	15.82	35.31	28.84	69.70	49.31
$h=2^{-9}$	32.97	35.25	66.28	61.01	139.62	112.22	306.71	201.86
(d) Total costs in seconds								

Appendix F. CPU times compared to an alternative smoother

Table F.15
CPU times (in s) for convergence with h-multigrid.

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	SCMS	ILUT	SCMS	ILUT	SCMS	ILUT	SCMS
$h=2^{-6}$	0.30	0.30	0.73	0.71	1.60	1.62	3.24	3.19
$h=2^{-7}$	1.13	1.19	2.90	2.86	6.42	6.47	12.78	11.73
$h=2^{-8}$	4.56	4.88	11.63	11.59	26.06	25.86	48.84	49.67
$h=2^{-9}$	19.08	19.44	46.63	46.19	95.92	104.10	202.52	202.64
(a) Assembly costs in seconds								
$h=2^{-6}$	0.08	0.01	0.25	0.01	0.63	0.02	1.02	0.02
$h=2^{-7}$	0.39	0.02	1.25	0.03	2.80	0.06	5.79	0.08
$h=2^{-8}$	1.66	0.09	5.71	0.12	14.23	0.22	27.66	0.27
$h=2^{-9}$	7.27	0.33	24.74	0.43	59.12	0.88	156.73	1.04
(b) Setup costs smoother in seconds								
$h=2^{-6}$	0.03	0.07	0.06	0.11	0.09	0.16	0.14	0.25
$h=2^{-7}$	0.08	0.22	0.11	0.34	0.23	0.58	0.37	0.80
$h=2^{-8}$	0.25	0.96	0.34	1.45	0.91	2.22	1.11	3.03
$h=2^{-9}$	0.88	4.34	1.17	5.72	3.19	9.11	3.85	12.13
(c) Solver costs in seconds								
$h=2^{-6}$	0.41	0.38	1.04	0.83	2.32	1.80	4.40	3.46
$h=2^{-7}$	1.60	1.43	4.26	3.23	9.45	7.11	18.94	12.61
$h=2^{-8}$	6.47	5.93	17.68	13.16	41.20	28.30	77.61	52.97
$h=2^{-9}$	27.23	24.11	72.54	52.34	158.23	114.09	363.10	215.81
(d) Total costs in seconds								

References

[1] T.J.R. Hughes, J.A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Comput. Methods Appl. Mech. Engrg. 194 (39–41) (2005) 4135–4195.

[2] T.J.R. Hughes, A. Reali, G. Sangalli, Duality and unified analysis of discrete approximations in structural dynamics and wave propagation: Comparison of p-method finite elements with k-method NURBS, Comput. Methods Appl. Mech. Engrg. 197 (49–50) (2008) 4104–4124.

[3] J.A. Cottrell, A. Reali, Y. Bazilevs, T.J.R. Hughes, Isogeometric analysis of structural vibrations, Comput. Methods Appl. Mech. Engrg. 195 (41–43) (2006) 5257–5296.

[4] Y. Bazilevs, V.M. Calo, Y. Zhang, T.J.R. Hughes, Isogeometric fluid–structure interaction analysis with applications to arterial blood flow, Comput. Mech. 38 (4–5) (2006) 310–322.

[5] W.A. Wall, M.A. Frenzel, C. Cyron, Isogeometric structural shape optimization, Comput. Methods Appl. Mech. Engrg. 197 (33–40) (2008) 2976–2988.

[6] G. Sangalli, M. Tani, Isogeometric preconditioners based on fast solvers for the Sylvester equation, SIAM J. Sci. Comput. 38 (6) (2016) 3644–3671.

[7] L. Beirao da Veiga, D. Cho. L.F. Pavarino, S. Scacchi, Overlapping Schwarz methods for isogeometric analysis, SIAM J. Numer. Anal. 50 (3) (2012) 1394–1416.

[8] K.P.S. Gahalaut, J.K. Kraus, S.K. Tomar, Multigrid methods for isogeometric discretizations, Comput. Methods Appl. Mech. Engrg. 253 (2013) 413–425.

[9] M. Donatelli, C. Garoni, C. Manni, S. Capizzano, H. Speleers, Symbol-based multigrid methods for Galerkin B-spline isogeometric analysis, SIAM J. Numer. Anal. 55 (1) (2017) 31–62.

[10] C. Hofreither, S. Takacs, W. Zulehner, A robust multigrid method for isogeometric analysis in two dimensions using boundary correction, Comput. Methods Appl. Mech. Engrg. 316 (2017) 22–42.

[11] C. Hofreither, S. Takacs, Robust multigrid for isogeometric analysis based on stable splittings of spline spaces, SIAM J. Numer. Anal. 4 (55) (2017) 2004–2024.

[12] J. Sogn, S. Takacs, Robust multigrid solvers for the biharmonic problem in isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 77 (2019) 105–124.

[13] A. de la Riva, C. Rodrigo, F. Gaspar, An efficient multigrid solver for isogeometric analysis, 2018, arXiv:1806.05848v1.

[14] K.J. Fidkowski, T.A. Oliver, J. LU, D.L. Darmofal, p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations, J. Comput. Phys. 207 (1) (2005) 92–113.

[15] H. Luo, J.D. Baum, R. Löhner, A p-multigrid discontinuous Galerkin method for the Euler equations on unstructured grids, J. Comput. Phys. 211 (2) (2006) 767–783.

[16] H. Luo, J.D. Baum, R. Löhner, Fast p-Multigrid discontinuous Galerkin method for compressible flows at all speeds, AIAA J. 46 (3) (2008) 635–652.

[17] P. van Slingerland, C. Vuik, Fast linear solver for diffusion problems with applications to pressure computation in layered domains, Comput. Geosci. 18 (3–4) (2014) 343–356.

[18] B. Helenbrook, D. Mavriplis, H. Atkins, Analysis of p-multigrid for continuous and discontinuous finite element discretizations, in: 16th AIAA Computational Fluid Dynamics Conference, Fluid Dynamics and Co-located Conferences, 2003.

[19] R. Tielen, M. Möller, C. Vuik, Efficient multigrid based solvers for Isogeometric Analysis, in: Proceedings of the 6th European Conference on Computational Mechanics and the 7th European Conference on Computational Fluid Dynamics, Glasgow, UK, 2018.

[20] Y. Saad, ILUT: A dual threshold incomplete LU factorization, Numer. Linear Algebra Appl. 1 (4) (1994) 387–402.

[21] C. Gianelli, B. Jüttler, H. Speleers, THB-splines: The truncated basis for hierarchical splines, Comput. Aided Geom. Design 29 (7) (2012) 485–498.

[22] C. De Boor, A Practical Guide to Splines, first ed., Springer-Verlag, New York, 1978.

[23] A. Brandt, Multi-level adaptive solutions to boundary-value problems, Math. Comp. 31 (138) (1977) 333–390.

[24] W. Hackbush, Multi-Grid Methods and Applications, Springer, Berlin, 1985.

[25] R. Tielen, M. Möller, C. Vuik, A direct projection to low-order level for p-multigrid methods in Isogeometric Analysis, in: The Proceedings of the European Numerical Mathematics and Advanced Applications Conference, Egmond aan Zee, the Netherlands, 2019 accepted for publication.

[26] S.C. Brenner, L.R. Scott, The Mathematical Theory of Finite Element Methods, in: Texts Applied Mathematics, vol. 15, Springer, New York, 1994.

[27] W.L. Briggs, V.E. Henson, S.F. McCormick, A Multigrid Tutorial, second ed., SIAM, Philadelphia, 2000.

[28] R.S. Sampath, G. Biros, A parallel geometric multigrid method for finite elements on octree meshes, SIAM J. Sci. Comput. 32 (3) (2010) 1361–1392.

[29] L. Gao, V. Calo, Fast isogeometric solvers for explicit dynamics, Comput. Methods Appl. Mech. Engrg. 274 (2014) 19–41.

[30] N. Collier, L. Dalcin, D. Pardo, V. Calo, The costs of continuity: performance of iterative solvers on isogeometric finite elements, SIAM J. Sci. Comput. 35 (2) (2013) 767–784.

[31] G. Guennebaud, J. Benoît, et al., Eigen v3, 2010, http://eigen.tuxfamily.org.

[32] Y. Saad, SPARSKIT: a basic tool kit for sparse matrix computations, 1994, http://www.cs.umn.edu/Research/arpa/SPARSKIT/sparskit.html.

[33] P.R. Amestoy, T.A. Davis, I.S. Duff, An approximate minimum degree ordering algorithm, SIAM J. Matrix Anal. Appl. 17 (4) (1996) 886–905.

[34] F. Calabro, G. Sangalli, M. Tani, Fast formation of isogeometric Galerkin matrices by weighted quadrature, Comput. Methods Appl. Mech. Engrg. 316 (2017) 606–622.
[35] P. Antolin, A. Buffa, F. Calabrò, M. Martinelli, G. Sangalli, Efficient matrix computation for tensor-product isogeometric analysis: The use of sum factorization, Comput. Methods Appl. Mech. Engrg. 285 (2015) 817–828.
[36] A. Mantzaflaris, B. Jüttler, B.N. Khoromskij, U. Langer, Low rank tensor methods in Galerkin-based isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 316 (2017) 1062–1085.
[37] C. Hofreither, W. Zulehner, Spectral analysis of geometric multigrid methods for isogeometric analysis, Numer. Methods Appl. 8962 (2015) 123–129.
[38] U. Trottenberg, C. Oosterlee, A. Schüller, Multigrid, Academic Press, 2001.
[39] G+Smo (Geometry plus Simulation modules), http://github.com/gismo.
[40] S. Takacs, Robust approximation error estimates and multigrid solvers for isogemeotric multi-patch discretizations, Math. Models Methods Appl. Sci. 28 (10) (2018) 1899–1928.
[41] G.B. Diaz Cortes, C. Vuik, J.D. Jansen, On POD-based Deflation Vectors for DPCG applied to porous mediaproblems, J. Comput. Appl. Math. 330 (2018) 193–213.
[42] R. Kraft, Adaptive and linearly independent multilevel B-splines, in: A. Le Méhauté, C. Rabut, L.L. Schumaker (Eds.), Surface Fitting and Multiresolution Methods, Vanderbilt University Press, Nashville, 1997, pp. 209–218.
[43] A. Vuong, C. Giannelli, B. Jüttler, B. Simeon, A hierarchical approach to adaptive local refinement in isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 200 (2011) 3554–3567.
[44] C. Hofreither, B. Jüttler, G. Kiss, W. Zulehner, Multigrid methods for isogeometric analysis with THB-splines, Comput. Methods Appl. Mech. Engrg. 308 (2016) 96–112.
[45] C. Bracco, D. Cho, C. Giannelli, R. Vazquez, BPX preconditioners for isogeometric analysis using (truncated) hierarchical B-spline, arXiv:1912.12073 [math.NA].
[46] C. Hofreither, L. Mitter, H. Speleers, Local Multigrid Solvers for Adaptive Isogeometric Analysis in Hierarchical Spline Spaces, NuMa-Report No. 2019–05, 2019.