A comparison of block preconditioners for isogeometric analysis discretizations of the incompressible Navier–Stokes equations

Horníková, Hana; Vuik, Cornelis; Egermaier, Jiří

DOI
10.1002/fld.4952

Publication date
2020

Document Version
Final published version

Published in
International Journal for Numerical Methods in Fluids

Citation (APA)
Horníková, H., Vuik, C., & Egermaier, J. (2020). A comparison of block preconditioners for isogeometric analysis discretizations of the incompressible Navier–Stokes equations. *International Journal for Numerical Methods in Fluids*, *93*(6), 1788-1815. https://doi.org/10.1002/fld.4952

Received: 14 January 2020 | Revised: 23 October 2020 | Accepted: 13 December 2020

DOI: 10.1002/fld.4952

RESEARCH ARTICLE

WILEY

A comparison of block preconditioners for isogeometric analysis discretizations of the incompressible Navier–Stokes equations

Hana Horníková[1] | **Cornelis Vuik**[2] | **Jiří Egermaier**[1]

[1]Faculty of Applied Sciences, University of West Bohemia, Pilsen, Czech Republic

[2]Delft Institute of Applied Mathematics, Delft University of Technology, Delft, The Netherlands

Correspondence
Hana Horníková, Faculty of Applied Sciences, University of West Bohemia, Univerzitní 8, 301 00 Pilsen, Czech Republic.
Email: hhornik@kma.zcu.cz

Funding information
Grantová Agentura České Republiky, Grant/Award Number: 19-04006S

Abstract

We deal with numerical solution of the incompressible Navier–Stokes equations discretized using the isogeometric analysis (IgA) approach. Similarly to finite elements, the discretization leads to sparse nonsymmetric saddle-point linear systems. The IgA discretization basis has several specific properties different from standard FEM basis, most importantly a higher interelement continuity leading to denser matrices. We are interested in iterative solution of the resulting linear systems using a Krylov subspace method (GMRES) preconditioned with several state-of-the-art block preconditioners. We compare the efficiency of the ideal versions of these preconditioners for three model problems (for both steady and unsteady flow in two and three dimensions) and investigate their properties with focus on the IgA specifics, that is, various degree and continuity of the discretization basis. Our experiments show that the block preconditioners can be successfully applied to the systems arising from high continuity IgA, moreover, that the high continuity can bring some benefits in this context. For example, some of the preconditioners, whose convergence is h-dependent in the steady case, seem to be less sensitive to the mesh refinement for higher continuity discretizations. In the unsteady case, we generally get faster convergence for higher continuity than for C^0 continuous discretizations of the same degree for most of the preconditioners.

KEYWORDS

block preconditioners, incompressible flow, isogeometric analysis, linear solvers, Navier–Stokes

1 | INTRODUCTION

Isogeometric analysis (IgA) is a relatively new discretization approach,[1,2] which has many in common with the finite element method (FEM). Its main goals are exact representation of the geometry even for coarse discretizations and simplification of the mesh refinement procedure. Motivated by the fact that the computational domains in industrial practice are usually B-spline/NURBS-based objects designed in CAD (Computer Aided Design) systems, the B-spline/NURBS basis is used for representing the approximate solutions in IgA.

Our work is inspired by an industrial application of the shape optimization of a water turbine runner blade. The goal is to improve the turbine efficiency in a wide range of operating conditions by changing the runner blade shape

automatically, such that the turbine development costs and time are reduced. The optimization problem is stated as minimization of a given objective function subject to constraints given by RANS equations (or the incompressible Navier–Stokes equations in the simplified laminar case). We are interested in IgA discretization of the governing equations for several reasons. First, the exact representation of the blade geometry is very important, since a piecewise polynomial approximation can cause spurious oscillations of the solution near the blade and in this application, a good resolution of the pressure on the blade is especially important. Also, the IgA approach is suitable for the purpose of automatic shape optimization, because it does not require new mesh generation every time the domain shape changes, since the computational mesh is already built in the geometry representation.

In this paper, we limit ourselves to the laminar flow described by the incompressible Navier–Stokes equations. Similarly to FEM, the IgA discretization of the linearized problem leads to a sequence of sparse saddle-point linear systems. One of the main differences between IgA and FEM is that the FEM solution is always C^0 continuous across the element boundaries; however, the higher degree IgA solution is usually of higher continuity. This leads to denser matrices, which makes the linear systems more expensive to solve compared to the linear systems of the same size arising from FEM discretizations.[3,4] However, Hughes et al.[2] show that for elliptic problems, the IgA solution of degree k has the same order of convergence as the classical FEM solution with basis functions of the same degree, independently of the order of continuity. The order of convergence describes, how the error of the approximate solution changes under mesh refinement. Assuming that something similar holds also for other than elliptic problems, we expect IgA to be more efficient than the classical FEM, since the number of degrees of freedom grows much slower with mesh refinement for IgA with basis functions of high continuity than for FEM. In other words, we expect to get an equally "good" solution for much less degrees of freedom using IgA.

We are interested in iterative solution methods for the saddle-point linear systems resulting from an IgA discretization of the linearized incompressible Navier–Stokes equations using preconditioned Krylov subspace methods. The key ingredient to the efficient iterative solution is a good preconditioner. There are generally two approaches to its construction – one is purely algebraic and the other is based on knowledge of the problem origin, discretization, matrix structure etc. The first involves preconditioners based on, for example, incomplete factorizations or sparse approximate inverses. In principle, these algebraic preconditioners can be applied in a black-box manner, but they often show poor convergence for saddle-point systems.[5] Moreover, the incomplete LU preconditioner can break down when applied to a saddle-point problem due to zero pivots if no fill-in is allowed. A suitable a priori renumbering of unknowns can be used to avoid zero pivots, decrease the memory and time requirements and also to improve the convergence of ILU preconditioned solvers.[6,7]

In this work, we focus on the so-called block preconditioners belonging to the second group of preconditioners, which exploit the knowledge of the system matrix structure and the physics behind individual blocks. Their derivation is based on a block LDU decomposition of the system matrix, splitting the problem into the velocity and pressure part. We consider two classes of block preconditioners – block triangular preconditioners based on approximation of the pressure Schur complement and SIMPLE-type preconditioners.

An example of a block triangular preconditioner is the pressure convection-diffusion preconditioner (PCD) proposed by Kay et al.,[8] where the Schur complement approximation is derived based on fundamental solution operators, and Silvester et al.[9] based on the idea of approximate commutators. A disadvantage of this method is the need to assemble a new matrix, a discrete convection-diffusion operator on the pressure space, which is not available in the standard FEM codes. To overcome this, Elman et al.[10] developed the least-squares commutator (LSC) preconditioner, where the PCD operator is computed from a (weighted) least-squares problem. It corresponds to a different interpretation of the BFBt preconditioner proposed earlier by Elman[11] or its scaled variant. Another approach was proposed by Benzi and Olshanskii.[12] They start with an augmented Lagrangian (AL) formulation of the saddle-point problem and construct a block triangular preconditioner, which is applied to the augmented system instead of the original one.

The Semi-Implicit Method for Pressure Linked Equations (SIMPLE) algorithm is a popular solver for incompressible flow problems. Vuik et al. proposed a Krylov accelerated version of SIMPLE-type methods[13,14] giving a recipe how to use these algorithms as block preconditioners for Krylov subspace methods. For steady problems, they show that it can result in much faster convergence to the steady state than if SIMPLE is used as a solver.[15]

These block preconditioners have been developed and successfully used for finite element discretizations of the incompressible Navier–Stokes equations. The question we try to answer in this paper is whether they are applicable to the IgA discretizations of high degree and high order of continuity with the same success. A similar study was done for IgA discretizations of the Stokes equations by Côrtes et al.,[16] where the authors combine a block triangular strategy with several "black-box" solvers to get a scalable preconditioner. We consider the steady and unsteady Navier–Stokes equations linearized by Picard method and present a comparison of ideal versions of several block preconditioners.

We dealt with this topic already in an earlier paper,[17] but there we focused rather on the basic properties of the preconditioners (mesh and Reynolds number dependence) and some specific aspects of flow in the water turbines. In the current paper, we systematically investigate their properties from the perspective of IgA specifics, that is, various degree and continuity of the discretization basis. The experiments are performed for three model problems: a flow over a two- (2D) and three-dimensional (3D) backward facing step and a flow around a 2D blade profile in a blade row.

The outline of this work is the following. In Section 2, the incompressible Navier–Stokes problem is formulated and its linearization and discretization are described with focus on the IgA approach. Section 3 gives a brief overview of the selected block preconditioners. In Section 4, the results of the numerical experiments for the test problems are presented. Finally, we give our conclusions in Section 5.

2 | PROBLEM FORMULATION

The mathematical model of incompressible viscous Newtonian flow is based on the incompressible Navier–Stokes equations (NSE). Let $\Omega \subset \mathbb{R}^d$ be a bounded domain, d being the number of spatial dimensions, with the boundary $\partial\Omega$ consisting of two complementary parts, Dirichlet $\partial\Omega_D$ and Neumann $\partial\Omega_N$. The steady-state incompressible Navier–Stokes problem is given as a system of $d+1$ differential equations together with boundary conditions

$$\begin{aligned} -\nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p &= \mathbf{0} && \text{in } \Omega, \\ \nabla\cdot\mathbf{u} &= 0 && \text{in } \Omega, \\ \mathbf{u} &= \mathbf{g}_D && \text{on } \partial\Omega_D, \\ \nu\frac{\partial\mathbf{u}}{\partial\mathbf{n}} - \mathbf{n}p &= \mathbf{0} && \text{on } \partial\Omega_N, \end{aligned} \tag{1}$$

where $\mathbf{u}$ is the flow velocity, p is the kinematic pressure, ν is the kinematic viscosity and $\mathbf{g}_D$ is a given function. The condition $\nu\frac{\partial\mathbf{u}}{\partial\mathbf{n}} - \mathbf{n}p = \mathbf{0}$ represents the classical "do-nothing" boundary condition. It does not have a physical meaning, but it is suitable at artificial outflow boundaries when modeling flows through a truncated domain, assuming that the physical domain continues further (for details, we refer to the References 18,19, where different outflow boundary conditions for such problems are discussed).

The Reynolds number is a dimensionless quantity expressing the ratio of inertial forces to viscous forces used to characterize the flow regime (laminar or turbulent). It is defined as

$$Re = \frac{UL}{\nu}, \tag{2}$$

where L is a characteristic length scale for the domain Ω and U is a reference velocity. The characteristic length scale L is chosen by convention for specific types of domain geometries.

2.1 | Galerkin discretization

In order to derive the weak formulation of the problem (1), we define velocity solution space V and test function space V_0 as follows

$$\begin{aligned} V &= \{\mathbf{u} \in H^1(\Omega)^d \mid \mathbf{u} = \mathbf{g}_D \text{ on } \partial\Omega_D\}, \\ V_0 &= \{\mathbf{v} \in H^1(\Omega)^d \mid \mathbf{v} = \mathbf{0} \text{ on } \partial\Omega_D\}. \end{aligned} \tag{3}$$

By multiplying the first line of (1) by a test function $\mathbf{v} \in V_0$ and the second line by a test function $q \in L^2(\Omega)$, integrating over Ω and using Green's theorem, we obtain the weak formulation: find $\mathbf{u} \in V$ and $p \in L^2(\Omega)$ such that

$$\nu\int_\Omega \nabla\mathbf{u} : \nabla\mathbf{v} + \int_\Omega (\mathbf{u}\cdot\nabla\mathbf{u})\cdot\mathbf{v} - \int_\Omega p\nabla\cdot\mathbf{v} = 0, \tag{4}$$

$$\int_\Omega q\nabla\cdot\mathbf{u} = 0, \tag{4}$$

for all $\mathbf{v} \in V_0$ and $q \in L^2(\Omega)$.

Because of the nonlinearity in the convective term, the problem needs to be linearized and solved iteratively. Using Picard's iteration method, we search for $\mathbf{u}^{n+1} \in V$ and $p^{n+1} \in L^2(\Omega)$ as a solution of the following Oseen problem

$$\nu \int_\Omega \nabla \mathbf{u}^{n+1} : \nabla \mathbf{v} + \int_\Omega (\mathbf{u}^n \cdot \nabla \mathbf{u}^{n+1}) \cdot \mathbf{v} - \int_\Omega p^{n+1} \nabla \cdot \mathbf{v} = 0, \tag{5}$$

$$\int_\Omega q \nabla \cdot \mathbf{u}^{n+1} = 0, \tag{5}$$

for all $\mathbf{v} \in V_0$ and $q \in L^2(\Omega)$. The initial guess $\mathbf{u}^0$ is often chosen to be zero, which means solving the Stokes equations in the first Picard iteration.

The idea of Galerkin discretization method is to define finite dimensional subspaces $V^h \subset V$, $V_0^h \subset V_0$, $W^h \subset L^2(\Omega)$ together with their basis functions and solve the problem (5) projected into these subspaces. Particular Galerkin-based discretization methods are then defined by the choice of the subspaces and their basis functions. After discretization, we obtain a nonsymmetric saddle-point linear system of the form

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}. \tag{6}$$

The matrix $F \in \mathbb{R}^{d \cdot n_u \times d \cdot n_u}$ is block diagonal,

$$F = \operatorname{diag}\left(\underbrace{A + N(\mathbf{u}_h^n), \ \dots \ , A + N(\mathbf{u}_h^n)}_{d} \right), \tag{7}$$

where the diagonal blocks consist of the discretization of the viscous term and the linearized convective term and $\mathbf{u}_h^n$ is the approximate velocity from the previous Picard iteration. $B^T \in \mathbb{R}^{d \cdot n_u \times n_p}$ and $B \in \mathbb{R}^{n_p \times d \cdot n_u}$ are discrete gradient and negative divergence operators, respectively, and n_u, n_p denote the number of velocity and pressure unknowns. Let us write the system (6) as

$$\mathcal{A}x = b, \tag{8}$$

for convenience. The right-hand side b comes from the Dirichlet boundary conditions where the coefficients of the basis functions are assumed to be known.

Note that in the case of a time-dependent problem, we discretize the time derivative of the velocity first (e.g., using an implicit time-stepping procedure), which leads to a sequence of spatial problems that can be discretized similarly to the steady-state problem. The discretization of the time-dependent problem leads to linear systems of the same form as (6), where the consistent velocity mass matrix multiplied by the reciprocal of time step appears in the block F.

To get a stable Galerkin discretization of a saddle-point problem, the so called LBB (or inf-sup) condition[20]

$$\inf_{q \in W^h} \sup_{\mathbf{v} \in V^h} \frac{\int_\Omega q \nabla \cdot \mathbf{v}}{||\mathbf{v}||_{H^1(\Omega)^d} ||q||_{L^2(\Omega)}} \geq \beta > 0, \tag{9}$$

has to be satisfied, where β is a constant independent of the mesh. Otherwise, a stabilization has to be used, which would introduce a nonzero block at the (2,2) position of the system matrix in (6). Here we consider only the case of so-called inf-sup stable combinations of velocity and pressure spaces ensuring that the inf-sup condition is satisfied. In standard finite elements, there are several known types of stable elements. One of the most popular is the Taylor–Hood element, where basis functions of degree k are used to approximate the pressure and basis functions of degree $k+1$ are used for the velocity components.

2.2 | Isogeometric analysis

In industrial practice, the computational domain Ω (or its parts) is usually a complex shape designed in a CAD system and represented using B-spline or NURBS (Non-Uniform Rational B-Spline) objects. For these domains, creating analysis suitable meshes is nontrivial and often very time consuming. IgA is a recently developed discretization approach based on Galerkin method aiming at integrating finite element analysis and CAD tools.[1,2] The basic idea is to use the data from the CAD model directly for analysis, it means to skip the meshing step and use the basis functions representing the geometry as basis functions of the solution space. In this section, we present a brief introduction to IgA. We limit ourselves only to domains described as B-spline objects and we do not go into details of NURBS.

Consider a partition of an interval $[a, b] \subset \mathbb{R}$ into ℓ subintervals determined by a vector $\mathbf{x} = (x_0, x_1, \dots, x_\ell)$, where $a = x_0 < x_1 < \dots < x_{\ell-1} < x_\ell = b$. Further consider a nonnegative integer k and a vector of regularities $\mathbf{r} = (r_1, r_2, \dots, r_{\ell-1})$. The spline space $S_k^{\mathbf{r}}$ is space of piecewise polynomial functions $f : [a, b] \to \mathbb{R}$ that satisfy the following conditions:

- f is a polynomial of degree at most k in $[x_i, x_{i+1})$ for $i = 0, \dots, \ell - 1$,
- f has r_i continuous derivatives at x_i for $i = 1, \dots, \ell - 1$.

A classical basis of the spline space $S_k^{\mathbf{r}}$ are the B-splines. The B-spline basis is defined by the spline degree k and a so-called knot vector Ξ. The knot vector is obtained from the vector $\mathbf{x}$ by repeating x_i such that its multiplicity is $m_i = k - r_i$ for $i = 1, \dots, \ell - 1$. In IgA, we usually work with open knot vectors, where the first and last knot are repeated $(k+1)$ times which corresponds to no continuity conditions at the endpoints of the interval $[a, b]$. Thus, the knot vector takes the form

$$\begin{aligned}\Xi &= (\underbrace{x_0, \dots, x_0}_{k+1}, \underbrace{x_1, \dots, x_1}_{k-r_1}, \dots, \underbrace{x_{\ell-1}, \dots, x_{\ell-1}}_{k-r_{\ell-1}}, \underbrace{x_\ell, \dots, x_\ell}_{k+1}) \\ &= (\xi_1, \xi_2, \dots, \xi_{n+k+1}),\end{aligned} \tag{10}$$

where $\xi_i \le \xi_{i+1}$ are the knots and n is the dimension of $S_k^{\mathbf{r}}$. The ith B-spline basis function of degree k, $N_{i,k}(\xi)$, $i = 1, 2, \dots, n$, is defined recursively as follows

$$\begin{aligned}N_{i,0}(\xi) &= \begin{cases} 1, & \xi_i \le \xi < \xi_{i+1}, \\ 0, & \text{otherwise}, \end{cases} \quad \text{for } k = 0, \\ N_{i,k}(\xi) &= \frac{\xi - \xi_i}{\xi_{i+k} - \xi_i} N_{i,k-1}(\xi) + \frac{\xi_{i+k+1} - \xi}{\xi_{i+k+1} - \xi_{i+1}} N_{i+1,k-1}(\xi), \quad \text{for } k > 0.\end{aligned} \tag{11}$$

Multivariate spline functions are defined by tensor-product of univariate splines. For example, given two nonnegative integers k, l, two vectors of regularities $\mathbf{r}_1, \mathbf{r}_2$ and the corresponding knot vectors $\Xi = (\xi_1, \xi_2, \dots, \xi_{n+k+1})$, $\Psi = (\psi_1, \psi_2, \dots, \psi_{m+l+1})$, the space of bivariate spline functions is defined as

$$S_{k,l}^{\mathbf{r}_1, \mathbf{r}_2} = S_k^{\mathbf{r}_1} \otimes S_l^{\mathbf{r}_2}, \tag{12}$$

with tensor-product B-spline basis functions $N_{i,k}(\xi) N_{j,l}(\psi)$. For more details on splines, we refer, for example, to de Boor.[21]

The B-spline basis can be used to describe objects like curves, surfaces, volumes, etc., in d-dimensional space $\mathbb{R}^d$. For example, a B-spline curve of degree k is constructed as a linear combination of the B-spline basis functions

$$\mathbf{C}(\xi) = \sum_{i=1}^{n} \mathbf{P}_i N_{i,k}(\xi), \tag{13}$$

where the coefficients $\mathbf{P}_i \in \mathbb{R}^d$, $i = 1, \dots, n$, are called control points.

The computational domain Ω can be represented by one or several B-spline objects, so-called patches. Let us consider a two-dimensional domain $\Omega \subset \mathbb{R}^2$ described as a single B-spline patch $\mathbf{G}(\xi, \psi)$ of degree k in both parameters over a

parametric domain $\hat{\Omega}$ defined by given knot vectors Ξ, Ψ,

$$\mathbf{G}(\xi,\psi) = \sum_{i=1}^{n}\sum_{j=1}^{m}\mathbf{P}_{i,j}N_{i,k}(\xi)N_{j,k}(\psi) = \sum_{i=1}^{n}\sum_{j=1}^{m}\mathbf{P}_{i,j}Q_{i,j}(\xi,\psi),$$
$$\mathbf{G} : (\xi,\psi) \mapsto (x,y), \quad (\xi,\psi) \in \hat{\Omega}, \quad (x,y) \in \Omega, \tag{14}$$

where we denote the product of two univariate B-spline basis functions of the same degree $N_{i,k}(\xi)N_{j,k}(\psi)$ as one bivariate function $Q_{i,j}(\xi,\psi)$. For simplicity of notation, let us re-index the control points and basis functions with one index r such that

$$\mathbf{G}(\xi,\psi) = \sum_{r=1}^{N}\mathbf{P}_r Q_r(\xi,\psi), \quad N = n \cdot m. \tag{15}$$

There is no need to generate any kind of mesh on the domain. The IgA elements are already defined by the knot vectors Ξ, Ψ in the parametric space. If the subsequent knots are different from each other, that is, $\xi_i \neq \xi_{i+1}$ and $\psi_j \neq \psi_{j+1}$, then $[\xi_i, \xi_{i+1}] \times [\psi_j, \psi_{j+1}]$ defines an element.

It is possible to refine the mesh easily (in the sense of tensor refinement) by inserting additional knots into the knot vector (h-refinement). After inserting new knots into the knot vector, new basis functions are formed using (11) and their control points are computed from the original control points. One can also elevate the polynomial degree of the basis (an analogue of the finite element p-refinement) without changing the geometry or its parametrization. This is done by increasing the multiplicity of all knots by one in order to preserve the order of continuity of the geometry and finding new control points for the obtained basis. Moreover, IgA offers a new refinement strategy referred to as k-refinement. It is a combination of knot insertion and degree elevation, where we first elevate the degree of the basis on the coarsest level and then insert new knots into the knot vector. This results in less-basis functions with less points of reduced order of continuity than if we first refine the knot vector and elevate the degree afterwards. See Hughes et al.[1] for more details on the refinement algorithms.

Let us now define the discrete spaces V^h, V_0^h, W^h as subspaces of V, V_0, W based on the basis functions Q_r defining Ω. Again, the velocity and pressure space must satisfy condition (9). The inf-sup stability of some combinations of the discrete solution spaces in IgA was shown in the literature.[22,23] As examples of stable combinations we name the isogeometric TH element (an analogue of the Taylor–Hood finite element) and the isogeometric subgrid (SG) element.[22] The isogeometric TH element can be defined by taking the pressure basis functions Q^p equal to the (h-, p- or k-refined) geometry basis functions and the velocity basis functions Q^u as the B-spline basis functions obtained from Q^p by degree elevation. Hence, the velocity basis functions are of degree $k+1$, both velocity and pressure bases are defined on the same mesh and have the same order of continuity. For the SG element, the velocity basis is defined on a subgrid of the pressure grid obtained by subdividing each pressure element into 2^d velocity elements in the sense of k-refinement, thus the velocity basis is generally of the maximum continuity C^k.

The discrete spaces on the physical domain Ω are generated by the push-forwards of the B-spline basis functions $Q_i^u(\xi,\psi)$ and $Q_j^p(\xi,\psi)$ on the parametric domain, that is,

$$Q_i^u(x,y) = Q_i^u(\xi,\psi) \circ \mathbf{G}^{-1}(x,y), \quad i = 1, \dots, N^u,$$
$$Q_j^p(x,y) = Q_j^p(\xi,\psi) \circ \mathbf{G}^{-1}(x,y), \quad j = 1, \dots, N^p, \tag{16}$$

where N^u, N^p are number of velocity and pressure basis functions, respectively. The discrete velocity $\mathbf{u}_h \in V^h$ and pressure $p_h \in W^h$ can be written as linear combinations of the basis functions

$$\mathbf{u}_h = \sum_{i=1}^{N_f^u}\mathbf{u}_i Q_i^u(x,y) + \sum_{i=N_f^u+1}^{N^u}\mathbf{u}_i^* Q_i^u(x,y), \tag{17}$$

$$p_h = \sum_{j=1}^{N^p} p_j Q_j^p(x,y), \tag{18}$$

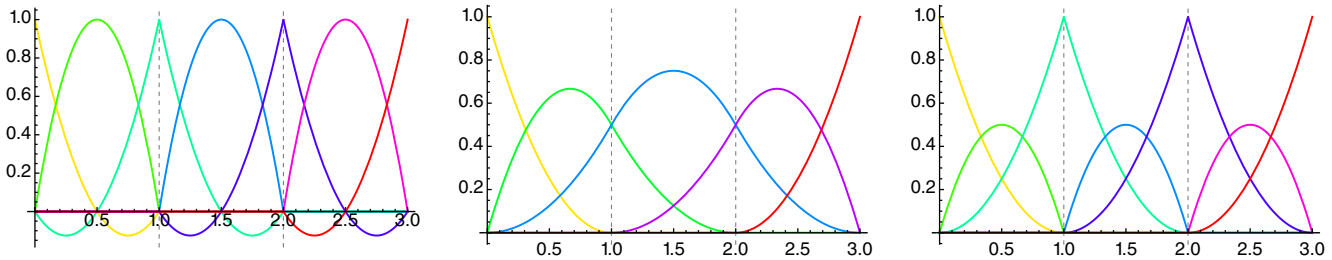

FIGURE 1 Comparison of one-dimensional quadratic discretization bases on a "mesh" with three elements: (A) finite element basis, (B) isogeometric basis with C^1 interelement continuity, (C) isogeometric basis with C^0 interelement continuity [Colour figure can be viewed at wileyonlinelibrary.com]

where $\mathbf{u}_i \in \mathbb{R}^2$ and $p_j \in \mathbb{R}$ are unknown coefficients and N_f^u is the number of "free" velocity basis functions whose coefficients are not fixed due to Dirichlet boundary conditions. The coefficients $\mathbf{u}_i^*$ come from a B-spline representation of the Dirichlet data at the boundary, which is assumed as known.

After substituting (17) and (18) into the linearized weak formulation, we obtain a linear system of the form (6) for the unknown coefficients $\mathbf{u}_i$ and p_j. The elements of the blocks A and $N(\mathbf{u}_h^n)$ are as follows

$$A = [A_{ij}] = \left[\nu \int_\Omega \nabla Q_i^u(x,y) \cdot \nabla Q_j^u(x,y) \mathrm{d}\Omega \right],$$
$$N(\mathbf{u}_h^n) = [N_{ij}(\mathbf{u}_h^n)] = \left[\int_\Omega Q_i^u(x,y) \left(\mathbf{u}_h^n \cdot \nabla Q_j^u(x,y) \right) \mathrm{d}\Omega \right]. \tag{19}$$

In the two-dimensional case, the matrix B consists of two blocks $B = [B_1, B_2]$, where

$$B_m = [B_{mij}] = \left[\int_\Omega Q_i^p(x,y) \left(\nabla Q_j^u(x,y) \cdot \mathbf{e}_m \right) \mathrm{d}\Omega \right] \quad \text{for } m = 1, 2. \tag{20}$$

Similarly to the FEM, the system matrix is sparse thanks to local supports of the B-spline basis functions. For degree $k = 0$ and $k = 1$ the B-spline basis is the same as the corresponding finite element basis. For higher degrees, however, they differ from each other. For illustration, see the comparison of quadratic FEM and IgA bases on a one-dimensional "mesh" consisting of three equal elements in Figure 1.

Figure 1(A) shows the standard quadratic FEM basis and Figure 1(B) shows a C^1 continuous quadratic B-spline basis for an open knot vector $\Xi = (0, 0, 0, 1, 2, 3, 3, 3)$. Obviously, we have less-basis functions for the same mesh in IgA, they are all pointwise nonnegative and generally not interpolatory, that is, their coefficients do not correspond to nodal values of the IgA solution. Moreover, their values form partition of unity for arbitrary $\xi \in (\xi_1, \xi_{n+k+1})$. The support of the B-spline basis function $N_{i,k}(\xi)$ is the interval $[\xi_i, \xi_{i+k+1}]$. Thus, the maximum number of B-spline basis functions for which the intersection of supports is not empty is $2k + 1$ (in one dimension). The same is true for FEM, hence both FEM and IgA basis result in sparse matrices of the same bandwidth. However, the FEM matrix is sparser since on each element there are basis functions with supports containing only that element, thus sharing support with only $k + 1$ basis functions. From the matrix structure point of view, a B-spline analogue of the FEM basis is a B-spline basis with C^0 interelement continuity displayed in Figure 1(C), which is obtained for the knot vector $\Xi = (0, 0, 0, 0, 1, 1, 2, 2, 3, 3, 3, 3)$.

3 | PRECONDITIONING TECHNIQUES

In this section we give a brief overview of several existing preconditioners for the saddle-point type problems of the form (6), developed usually in the context of finite element or finite volume discretizations, and we summarize some of their

Algorithm 1. Application of $\mathcal{M}_t^{-1}$

1: Solve $Sz_p = r_p$
2: Update $r_u = r_u - B^T z_p$
3: Solve $Fz_u = r_u$

properties known from the literature. We focus on a class of preconditioners called block preconditioners that are based on the block LDU decomposition of the system matrix

$$\mathcal{A} = \begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ BF^{-1} & I \end{bmatrix} \begin{bmatrix} F & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & F^{-1}B^T \\ 0 & I \end{bmatrix}, \tag{21}$$

where $S = -BF^{-1}B^T$ is the Schur complement matrix. We give only a basic description of the individual methods, for more details we refer to the literature.

3.1 | Block triangular preconditioners

The class of block triangular preconditioners is derived from an L(DU) coupling of the factors in (21), that is, from the following block decomposition

$$\mathcal{A} = \begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ BF^{-1} & I \end{bmatrix} \begin{bmatrix} F & B^T \\ 0 & S \end{bmatrix}. \tag{22}$$

Let us denote the upper block triangular factor as

$$\mathcal{M}_t = \begin{bmatrix} F & B^T \\ 0 & S \end{bmatrix}, \tag{23}$$

then the matrix $\mathcal{A}\mathcal{M}_t^{-1}$ is lower block triangular with identity diagonal blocks, hence with all eigenvalues equal to one. As shown by Murphy et al.,[24] the minimal polynomial of this matrix is of degree 2 and therefore, if we use $\mathcal{M}_t$ as a right preconditioner for GMRES, it will converge in at most two iterations. The same holds for left preconditioning.

The computation of the preconditioned residual $z = \mathcal{M}_t^{-1} r$ is performed by solving the linear system

$$\begin{bmatrix} F & B^T \\ 0 & S \end{bmatrix} \begin{bmatrix} z_u \\ z_p \end{bmatrix} = \begin{bmatrix} r_u \\ r_p \end{bmatrix} \tag{24}$$

in the steps summarized in Algorithm 1. Thus, it requires solving one linear system with the Schur complement matrix and one linear system with the matrix F.

In practice, we cannot use the matrix $\mathcal{M}_t$ as a preconditioner, since it would be too expensive. Especially, it is not practical to construct S explicitly, because it would require the explicit construction of F^{-1}, since it is multiplied with rectangular matrices from both sides, and S is a dense matrix. Therefore, we have to find some inexpensive approximation $\hat{S} \approx S$ first. The choice of the approximation yields different preconditioners.

The matrix F consists of d decoupled discrete convection-diffusion operators (in the case of Picard linearization of the Navier–Stokes equations), thus the solution of the linear system with F requires solution of several convection-diffusion subproblems. The application of the inverse approximate Schur complement $\hat{S}^{-1}$ often requires solution of subproblems with a Poisson-type discrete operator or mass matrix. If all of these subproblems are solved with a direct method, we talk about an ideal version of the preconditioner.

Since the use of direct methods is too expensive for large matrices and can become even unfeasible in the case of 3D problems, these subsystems have to be solved approximately. The approximate solvers for the subsystems should be optimal in the sense that their convergence is independent of the mesh parameter h and therefore multigrid methods (geometric or algebraic) are often used. An overview of efficient solution methods for finite element discretizations of flow problems, including multigrid methods for Poisson and convection-diffusion problems, can be found in Elman, Silvester, Wathen.[25] However, many authors in recent years have observed that the performance of classical multigrid methods applied to IgA linear systems is highly dependent on the B-spline degree and the spatial dimension. Several robust multigrid methods specialized for IgA were developed based on different ideas, usually exploiting the tensor-product structure of the spline spaces, see, for example, Donatelli et al.,[26] Hofreither and Takacs,[27] Hofer and Takacs,[28] or Riva et al.[29] Alternatively, p-multigrid, where "coarsening" in the spline degree is performed, is considered by Tielen et al.[30] These works are mainly focused on the Poisson problem, but some of them consider a more general convection-diffusion-reaction equation.[26,30]

Besides multigrid, one can also use a small number of iterations of a preconditioned iterative solver for the subsystems. There are some robust preconditioners developed specifically for IgA discretizations of the Poisson problem, see Harbrecht et al.[31] or Sangalli and Tani.[32]

In this work, we are interested in the performance of the block preconditioners applied to IgA discretizations of the Navier–Stokes equations without the influence of the inner solvers choice. Therefore, we consider only the ideal versions of the preconditioners presented in this section.

3.1.1 | Approximate commutator preconditioners

Recall that the matrix F is a discretization of the convection-diffusion operator defined on the velocity space, which can be written as

$$\mathcal{L} = -\nu\Delta + (\mathbf{w}\cdot\nabla) \quad \text{or} \quad \mathcal{L} = \frac{1}{\Delta t} - \nu\Delta + (\mathbf{w}\cdot\nabla), \tag{25}$$

for the steady or unsteady problem, respectively, where $\mathbf{w}$ is the approximation of the velocity computed in the most recent Picard iteration. Suppose that an analogous operator $\mathcal{L}_p$ is well defined on the pressure space. Two popular block triangular preconditioners, the PCD preconditioner and the LSC preconditioner, are based on the idea that the commutator with the gradient operator defined as

$$\mathcal{E} = \mathcal{L}\nabla - \nabla\mathcal{L}_p, \tag{26}$$

is small in some sense. In this section, we follow the exposition given in chapter 9 of Elman, Silvester, Wathen.[25]

The discrete version of the commutator $\mathcal{E}$ in terms of finite element matrices takes the form

$$E = (M_u^{-1}F)(M_u^{-1}B^T) - (M_u^{-1}B^T)(M_p^{-1}F_p), \tag{27}$$

where M_u and M_p are the velocity and pressure mass matrix, respectively, and F_p is a discrete version of $\mathcal{L}_p$. Assume that E is also small, which means that

$$(M_u^{-1}F)(M_u^{-1}B^T) \approx (M_u^{-1}B^T)(M_p^{-1}F_p). \tag{28}$$

After several algebraic manipulations, this leads to the following approximation to the Schur complement

$$S = -BF^{-1}B^T \approx -BM_u^{-1}B^TF_p^{-1}M_p. \tag{29}$$

In the PCD preconditioner, the matrix $BM_u^{-1}B^T$ is replaced by the spectrally equivalent discrete pressure Laplacian matrix A_p. The PCD Schur complement approximation then takes the form

$$\hat{S}_{\text{PCD}} = -A_pF_p^{-1}M_p. \tag{30}$$

The application of $\hat{S}_{\mathrm{PCD}}^{-1}$ requires one mass matrix solve, one Poisson solve and a matrix-vector multiplication with the matrix F_p.

Although eigenvalues of the preconditioned matrix alone do not determine the convergence of Krylov subspace methods for nonsymmetric systems, it is known that a well-clustered spectrum away from zero results in fast convergence in many cases. For finite element discretizations with quasi-uniform meshes, it has been shown in the literature (e.g., Loghin[33]) that the modulus of the eigenvalues of the matrix preconditioned with PCD is bounded (away from zero) independently of the mesh parameter h. Convergence of Krylov subspace methods independent of h is often observed, especially for enclosed flows, and mildly dependent on the Reynolds number. The main disadvantage of this preconditioner is the need to construct two extra discrete operators A_p and F_p, which are usually not readily available in standard finite element codes. Moreover, the choice of boundary conditions for these operators is not clear and a poor choice can significantly affect the performance of the preconditioner. In the original version of PCD, both operators are defined with Dirichlet conditions at the inflow part of the boundary and Neumann conditions at the rest of the boundary. Elman and Tuminaro[34] proposed an improved version of the preconditioner for coordinate-aligned domains, where F_p is defined with Robin boundary condition at the inflow part of the boundary. Note that the results reported in this paper are obtained for the original PCD preconditioner.

The LSC preconditioner avoids the explicit construction of the operators on pressure space and defines the jth column of F_p as a solution of the following weighted least-squares problem

$$\min || \, [M_u^{-1} F M_u^{-1} B^T]_j - M_u^{-1} B^T M_p^{-1} [F_p]_j \, ||_{M_u}, \tag{31}$$

where $||x||_{M_u} = \sqrt{x^T M_u x}$ is a discrete analogue of the continuous L^2 norm on the velocity space. The vector $[F_p]_j$ is obtained by solving the normal equations

$$M_p^{-1} B M_u^{-1} B^T M_p^{-1} [F_p]_j = [M_p^{-1} B M_u^{-1} F M_u^{-1} B^T]_j, \tag{32}$$

which leads to the following definition of F_p:

$$F_p = M_p (B M_u^{-1} B^T)^{-1} (B M_u^{-1} F M_u^{-1} B^T). \tag{33}$$

Substituting this into (29) we get the following approximation of the Schur complement

$$\hat{S}_{\mathrm{LSC}} = -(B M_u^{-1} B^T)(B M_u^{-1} F M_u^{-1} B^T)^{-1} (B M_u^{-1} B^T). \tag{34}$$

Here, the inverse of the velocity mass matrix in $B M_u^{-1} B^T$ is replaced by the inverse of its diagonal $\hat{M}_u$.

The application of $\hat{S}_{\mathrm{LSC}}^{-1}$ involves solving two subsystems with the matrix $B \hat{M}_u^{-1} B^T$, which is essentially a discrete Laplace operator. Thus, two Poisson-type solves and one matrix-vector product are needed. The convergence of LSC is generally dependent on both mesh size and Reynolds number. Its advantage is that it is built from available matrices and no additional boundary conditions have to be defined, although a modification taking special care of the boundary may enhance the convergence.[34]

3.1.2 | AL preconditioner

A different approach has been proposed by Benzi and Olshanskii.[12] The original system (6) is replaced with the equivalent system

$$\begin{bmatrix} F_\gamma & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f_\gamma \\ g \end{bmatrix}, \tag{35}$$

where $F_\gamma = F + \gamma B^T W^{-1} B, f_\gamma = f + \gamma B^T W^{-1} g, \gamma > 0$ is a parameter and W is a positive definite matrix. The system (35) is then preconditioned with the block triangular preconditioner

$$\mathcal{M}_{\mathrm{AL}} = \begin{bmatrix} F_\gamma & B^T \\ 0 & \hat{S}_{\mathrm{AL}} \end{bmatrix}, \tag{36}$$

where the inverse of the Schur complement approximation is given by

$$\hat{S}_{\mathrm{AL}}^{-1} := -\nu \tilde{M}_p^{-1} - \gamma W^{-1} \tag{37}$$

and $\tilde{M}_p$ is a pressure mass matrix approximation, usually a diagonal matrix. The matrix W is often chosen to be equal to $\tilde{M}_p$.

Of course, the choice of the parameter γ is important. A large value would lead to small number of iterations of the preconditioned Krylov method, but for large γ the block F_γ becomes increasingly ill-conditioned and makes the solution of the subsystems expensive.[6] Hence, it is often set $\gamma \approx 1$.

It has been shown for standard finite elements,[12] assuming that $W = \tilde{M}_p = M_p$, that the eigenvalues of the preconditioned matrix are contained in a rectangular region in the right half-plane of the complex plane independent of h and for $\gamma = O(\nu^{-1})$ also independent of ν. However, rapid h- and ν-independent convergence is observed also for smaller values of γ and even if the subsystem with F_γ is solved approximately. The solution of the subsystems with F_γ is the main difficulty of this approach. The additional term $\gamma B^T W^{-1} B$ makes the matrix less sparse compared to F and introduces a coupling between the velocity components which is not present in the discretization of the Picard linearization of the Navier–Stokes equations. Direct solution of these subsystems becomes very expensive and finding an effective approximate solver can be difficult. Benzi and Olshanskii[12] proposed a suitable multigrid method in two dimensions and their approach was later generalized to three dimensions.[35] However, these specialized multigrid methods are strongly tied with the discretization and currently limited to some particular FEM discretizations.

Modified version

One way to simplify the solution of the systems with F_γ is the modified version of AL preconditioner (MAL).[36] Let us denote the particular blocks of F_γ in two dimensions as follows

$$F_\gamma = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}. \tag{38}$$

The modified approach suggests to replace this block by its upper block triangle

$$\tilde{F}_\gamma =: \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}, \tag{39}$$

such that instead of solving the whole system at once, we solve two smaller systems with the blocks A_{11} and A_{22}. These blocks can be interpreted as discrete anisotropic convection-diffusion operators, thus, applying $\tilde{F}_\gamma^{-1}$ requires solving two anisotropic convection-diffusion problems. The situation is similar in three dimensions, where we have to solve three subsystems.

3.2 | SIMPLE-type preconditioners

SIMPLE is an algorithm for numerical solution of the Navier–Stokes equations developed for finite volume and finite difference discretizations by Patankar and Spalding.[37] It is based on decoupling the system of equations and solving the velocity and pressure part separately. First, the velocity is solved from the momentum equations assuming that the

Algorithm 2. SIMPLE algorithm

1: Solve $Fu^* = f - B^T p^*$
2: Solve $\hat{S}_S \delta p = g - Bu^*$
3: Update $u = u^* + \delta u$, where $\delta u = -D^{-1}B^T \delta p$
4: Update $p = p^* + \delta p$

pressure is known from the previous iteration. Then, the pressure and velocity are corrected in order to satisfy the discrete continuity equation.

The algorithm can be written in the form of block matrices. Starting from an (LD)U coupling of the factors in (21), that is, from the block decomposition

$$\mathcal{A} = \begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} = \begin{bmatrix} F & 0 \\ B & S \end{bmatrix} \begin{bmatrix} I & F^{-1}B^T \\ 0 & I \end{bmatrix}, \tag{40}$$

the SIMPLE algorithm is obtained by using the approximation $F^{-1} \approx D^{-1} = \mathrm{diag}(F)^{-1}$, introducing intermediate values u^*, p^* and corrections $\delta u, \delta p$ such that

$$u = u^* + \delta u, \quad p = p^* + \delta p, \tag{41}$$

and solving the system in the following two steps:

$$\begin{bmatrix} F & 0 \\ B & \hat{S}_S \end{bmatrix} \begin{bmatrix} u^* \\ \delta p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}, \tag{42}$$

and

$$\begin{bmatrix} I & D^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} u^* \\ \delta p \end{bmatrix}, \tag{43}$$

where $\hat{S}_S = -BD^{-1}B^T$. These steps are performed recursively, leading to the Algorithm 2, where the intermediate pressure p^* is estimated from the prior iterations.

This algorithm can be also seen as a stationary distributive iterative method for solving the system (6) (see Wesseling[38]) and based on that, the SIMPLE preconditioner can be derived.[13,14,39] The preconditioner matrix takes the form

$$\mathcal{M}_S = \begin{bmatrix} F & 0 \\ B & \hat{S}_S \end{bmatrix} \begin{bmatrix} I & D^{-1}B^T \\ 0 & I \end{bmatrix}, \tag{44}$$

and the application of $\mathcal{M}_S^{-1}$ corresponds to one iteration of the Algorithm 2 with $p^* = 0$.

There are several modifications of the algorithm. One of them is called SIMPLER, where p^* is obtained as a solution of the system

$$\hat{S}_S p^* = g - BD^{-1}((D - F)u^k + f), \tag{45}$$

where u^k is the velocity from the previous iteration if SIMPLER is used as a solver, but it is taken equal to zero in the case of preconditioner.

Another variant is the MSIMPLER algorithm, which is obtained from SIMPLER by replacing all occurrences of D by an approximation of the velocity mass matrix $\hat{M}_u$. The choice of $\hat{M}_u$ depends on the particular type of elements. In this paper, we choose $\hat{M}_u$ equal to the diagonal of M_u. For more details on this preconditioner, see Rehman et al.[39]

The convergence of all mentioned SIMPLE-type preconditioners is dependent on mesh size and Reynolds number. The SIMPLE preconditioner is cheap per iteration. One only has to solve one velocity subsystem and one pressure subsystem. The SIMPLER and MSIMPLER preconditioners need one additional pressure solve per iteration, but they generally give much faster convergence than SIMPLE. The advantage of MSIMPLER over SIMPLER is that the Schur complement approximation is constructed only once, since $\hat{M}_u^{-1}$ does not change during the linearization and time steps.

4 | NUMERICAL EXPERIMENTS

We present a comparison of convergence of the preconditioners mentioned in Section 3 for a well-known benchmark problem of flow in a simple backward facing step geometry in 2D and 3D and for a more complex 2D domain from industrial practice. We especially focus on specifics of the IgA discretizations like higher-degree basis functions and their interelement continuity. For all test problems, we compare the convergence for a set of discretizations with various degree and continuity and observe its dependence on the uniform and local mesh refinement. For the unsteady 2D backward facing step, we perform the experiments for three different Reynolds numbers.

4.1 | Experiments settings

For each test problem, we are interested in iterative solution of one linear system obtained from the discretization of the incompressible Navier–Stokes equations for a given mesh and discretization bases. For the experiments, we consider linear systems obtained after performing 5 Picard iterations in the steady case or five time steps in the unsteady case. We solve them using GMRES with no restarts with various preconditioners, we always start from a zero initial solution and stop when we reach relative residual norm smaller than 10^{-6}.

Note that if we use the preconditioned GMRES with zero initial solution vector to solve each linear system during the Picard iteration in the steady case, the GMRES iteration count settles on a constant value after a few Picard iterations. From our experience, it is already settled in the fifth Picard iteration. In the unsteady case, there are almost no fluctuations in the GMRES iteration count (with zero initial solution) during the computation. Of course, in practical computations, it would be beneficial to choose the initial solution for GMRES as the solution vector from the previous Picard iteration.

The linear systems are obtained from an in-house isogeometric incompressible flow solver implemented in C++ within a framework of the G+Smo library, an open-source C++ library that implements the concept of IgA (for more information, see the documentation[40]). We also implemented all tested preconditioners in the framework of G+Smo exploiting the available linear algebra tools that are mostly inherited from the Eigen library.[41] Note that the incompressible flow solver and the implementation of the preconditioners are not part of the G+Smo library at the moment.

In this paper, we are interested in the convergence behavior of the ideal versions of the preconditioners, which means that all subsystems are solved with a direct solver (sparse LU available in Eigen).

4.2 | Test problems

4.2.1 | Backward facing step

The first test problem is a flow over a 2D backward facing step of height $h = 1$ and parabolic inlet velocity with maximum of 1. We set the "do-nothing" boundary condition at the outlet boundary and zero velocity at the rest of the boundary. In the case of unsteady flow, we use time step $\Delta t = 10^{-2}$. For most of the experiments we consider the kinematic viscosity $\nu = 10^{-2}$. The characteristic length L in the definition of the Reynolds number (2) is often chosen equal to the step hight. Setting the characteristic velocity U equal to the maximum inlet velocity, the Reynolds number is

$$Re = \frac{UL}{\nu} = \frac{1}{10^{-2}} = 100. \tag{46}$$

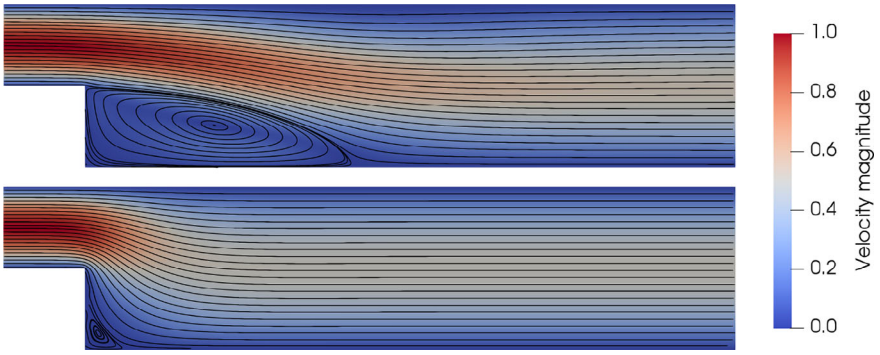

FIGURE 2 Velocity magnitude values and streamlines for the steady (top) and unsteady (bottom) problem with $Re = 100$ [Colour figure can be viewed at wileyonlinelibrary.com]

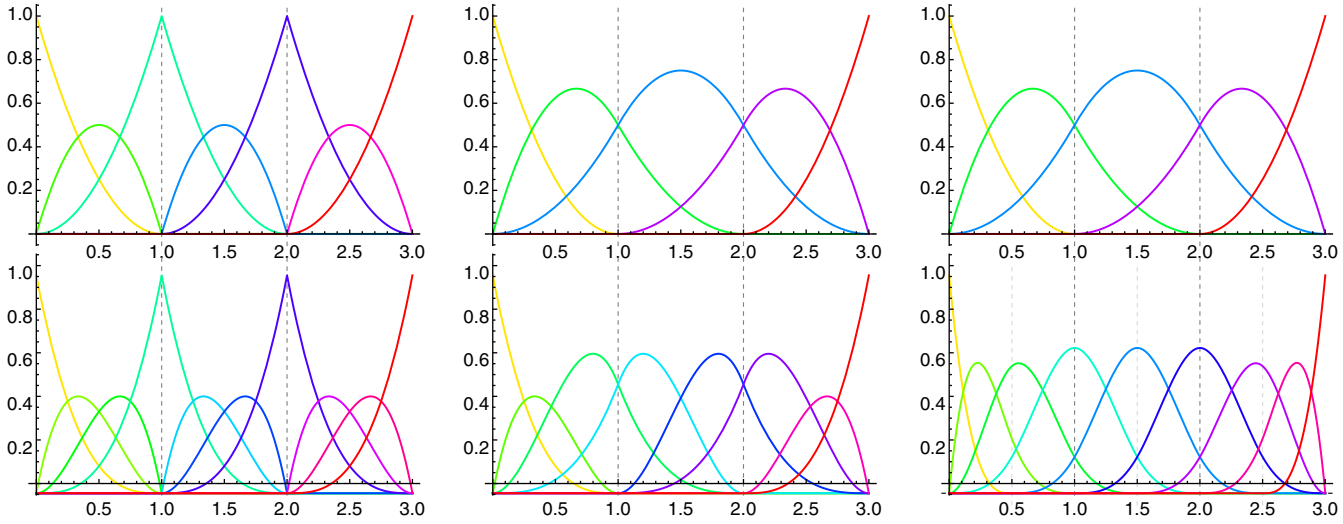

FIGURE 3 Examples of one-dimensional pressure (top) and velocity (bottom) discretization basis of degrees 2 and 3, respectively, with various order of continuity on a "mesh" with three elements: (a) isogeometric Taylor–Hood finite (TH) element with C^0 continuity, (B) isogeometric TH element with C^1 continuity, (c) isogeometric subgrid element with C^1 continuity for pressure and C^2 for velocity [Colour figure can be viewed at wileyonlinelibrary.com]

In the unsteady case, we also consider $Re = 1000$ and $Re = 10000$ for comparison. If not explicitly indicated otherwise, the results correspond to $Re = 100$.

Figure 2 shows the velocity with streamlines in the Picard iteration/time step where the linear systems in the experiments come from, in the steady/unsteady case, respectively.

The computational domain consists of three rectangular patches (see Figure 4) that can be described as B-spline surfaces of arbitrary degree k with control points forming an orthogonal grid in a plane. To investigate the influence of the discretization basis degree and its continuity properties on the convergence, we created several parametrizations of the domain using B-splines of degree k varying from 1 to 3 and various interelement continuity in the interior of the patches. We consider conforming meshes where the patches are "glued" together by identifying the corresponding basis functions at the interfaces, hence the continuity along the patch interfaces is always C^0. We use the two inf-sup stable combinations of velocity and pressure discretization spaces mentioned before, the isogeometric TH and SG element. For the TH element of degree k for pressure and $k+1$ for velocity we created discretizations of C^0 to C^{k-1} continuity. The C^0 continuous discretizations can be considered as an analogue of FEM discretization. Further, we consider the SG element with the highest possible continuity, that is, C^{k-1} for pressure and C^k for velocity. Figure 3 illustrates different combinations for quadratic pressure basis and cubic velocity basis.

Note that the backward facing step problem has a singularity – the pressure tends to infinity at the nonconvex corner. In practical computations, local refinement of the mesh in the vicinity of the corner is necessary to resolve the singularity. However, a very accurate solution for the backward facing step problem is not the main target of this paper. We are

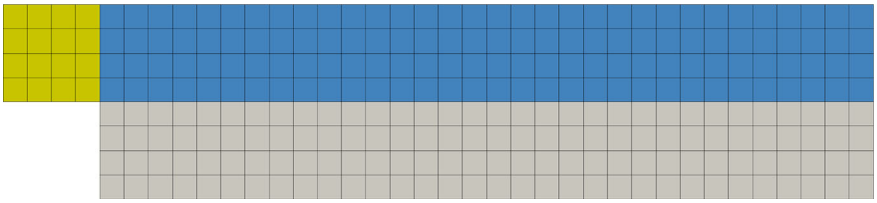

FIGURE 4 The backward facing step computational domain consisting of three patches, coarse uniform mesh M1 [Colour figure can be viewed at wileyonlinelibrary.com]

Num. elem.	M1	M2	M3
Step 2D	272	1088	4352
Step 3D	1152	9216	73,728
Profile 2D	288	1152	4608

TABLE 1 Number of elements of the initial mesh M1 and the meshes M2, M3 obtained from M1 by one and two uniform refinements, respectively, for all test problems

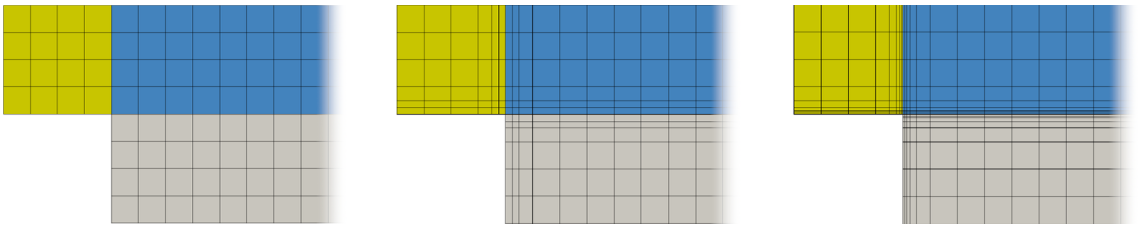

FIGURE 5 The coarse uniform mesh M1 (left) and two levels of local refinement (center and right) [Colour figure can be viewed at wileyonlinelibrary.com]

Step 2D (M3)	DOFs	nnz	nnz [%]
2-1, C^0	38,769	960,292	0.06%
3-2, C^0	95,233	3,968,584	0.04%
4-3, C^0	177,809	11,084,516	0.04%
3-2, C^1	40,378	2,346,622	0.14%
4-3, C^2	42,005	4,394,180	0.25%

TABLE 2 Number of degrees of freedom (DOFs), number of nonzero elements in the matrix and their percentage for various discretizations with the finest uniform mesh M3 of the 2D backward facing step

interested in the convergence properties of block preconditioners for different B-spline discretization spaces, such as their dependence on mesh refinement. For this purpose, we investigate three uniform meshes, the coarsest of which is shown in Figure 4 and the other two are obtained by one and two additional uniform refinements.

Let us call the three uniform meshes M1, M2, and M3 in the following. The numbers of elements of the three meshes are shown in Table 1. Further, we test stretched grids obtained from the uniform ones by recursively refining one row of elements near the non-convex corner. We consider two levels of local refinement shown in Figure 5.

Note that the size of the linear systems resulting from different discretizations varies even for the same mesh. Table 2 shows the number of degrees of freedom and number of nonzeros in the matrix for various discretizations with the mesh M3. In the C^0 case, the size of the matrix grows as the degree increases and the number of nonzeros also increases such that the density of the matrix stays the same. On the other hand, for increasing order of continuity together with the degree, the size of the matrix stays more or less constant, but the matrices become denser, thus the solution of the systems also becomes more expensive.

We perform some of the experiments also for the steady and time-dependent Navier–Stokes problem in a 3D backward facing step domain. The settings are similar as in 2D, that is, the kinematic viscosity $\nu = 10^{-2}$, time step $\Delta t = 10^{-2}$, inlet velocity with maximum magnitude of 1, the "do-nothing" outlet boundary condition and zero Dirichlet conditions for velocity at the rest of the boundary. The initial (coarse) uniform mesh M1 corresponds to the refinement displayed in Figure 4, but the domain is shorter. We again denote as M2 and M3 the meshes obtained by another one and two uniform

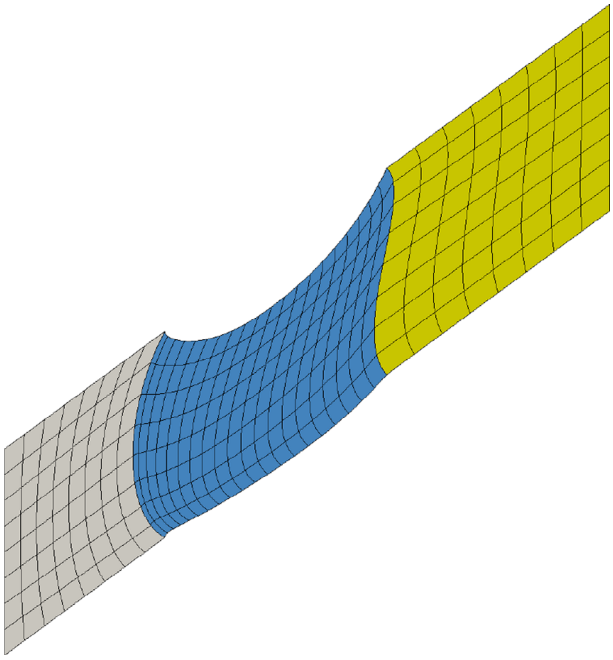

FIGURE 6 The computational domain between two parts of a blade profile [Colour figure can be viewed at wileyonlinelibrary.com]

refinements, respectively, and summarize their numbers of elements in Table 1. We consider the same set of discretizations of various degree and interelement continuity as in the 2D case.

4.2.2 | Turbine blade profile

The second test problem is flow in a domain that stems from a simplified problem of flow in a water turbine – a 2D blade row obtained by unfolding a cylindrical cross-section of the turbine. The computational domain is shown in Figure 6. It is a strip between two parts of a blade profile consisting of three B-spline patches of degree $k = 3$. The upper and lower boundary of the middle (blue) patch form the blade profile and periodic boundary conditions are set at the upper and lower boundaries of the remaining two patches.

We again consider a fluid with viscosity $\nu = 10^{-2}$. The inlet velocity (at the leftmost boundary) is a constant vector with a direction tangent to the camber line of the profile at the leading point and magnitude approximately 7.77. The Reynolds number based on the blade chord length and the inlet velocity magnitude is

$$Re = \frac{UL}{\nu} = \frac{7.77 \cdot 0.37}{10^{-2}} \doteq 287. \tag{47}$$

In the case of unsteady flow, we use time step $\Delta t = 10^{-4}$. The velocity magnitude values and streamlines in the fifth Picard iteration of the steady problem are shown in Figure 7, where two copies of the computational domain are displayed above each other. The unsteady solution looks similarly.

For this problem, we perform similar experiments as for the 2D backward facing step. We consider the natural choice of discretization bases based on the domain geometry description, that is, a C^2 continuous basis of degree 3 for pressure and a basis of degree 4 for velocity (either the isogeometric TH or SG element). This domain cannot be described exactly using B-splines of arbitrary degree and continuity. For comparison, we consider a linear approximation of the domain with the given mesh (see Figure 8 for illustration) with two different discretizations using the isogeometric TH element serving as analogues of low and high degree finite elements.

One of them uses linear pressure basis and quadratic velocity basis and the other uses a C^0 continuous cubic pressure basis obtained from the linear basis by degree elevation and a C^0 continuous quadratic velocity basis. Here, we denote as M1 the mesh displayed in Figure 6 and as M2, M3 the meshes obtained by one and two additional uniform refinements. The numbers of elements (shown in Table 1) are similar as for the 2D backward facing step with the corresponding refinement.

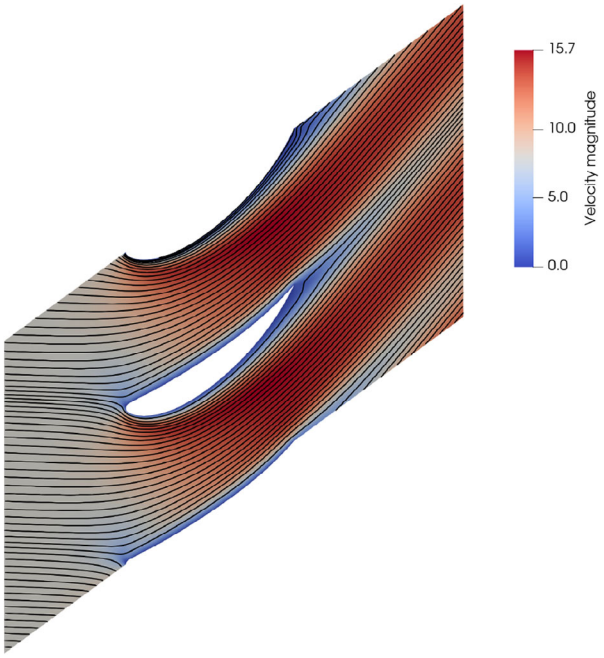

FIGURE 7 Velocity magnitude values and streamlines for the steady problem [Colour figure can be viewed at wileyonlinelibrary.com]

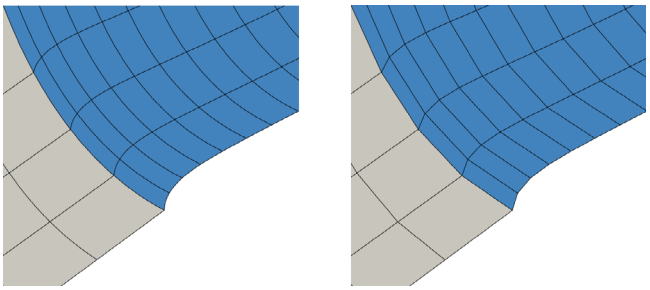

FIGURE 8 A detail of the original cubic domain (left) and its linear approximation (right) [Colour figure can be viewed at wileyonlinelibrary.com]

4.3 | Results in 2D

In this section, we present numerical results for the iterative solution of the linear systems arising from various discretizations of the 2D test problems. We present the numbers of GMRES iterations needed to reach relative residual norm smaller than 10^{-6} with the particular preconditioners for both steady and unsteady problems. The individual discretizations are denoted, for example,"2-1, C^0" meaning a C^0 continuous isogeometric TH element with quadratic basis for velocity and linear basis for pressure. As mentioned above, not all discretization spaces considered for the backward facing step geometry could be used also for the turbine blade profile geometry. Therefore, some rows of the Tables 3 and 5 corresponding to the profile geometry are filled with hyphens "-".

We do not mention MSIMPLER in the tables for brevity, since its convergence is very similar to LSC in all cases. For the backward facing step, they even differ only by one or two iterations for most discretizations. For the blade profile, MSIMPLER takes usually around 10 more iterations than LSC in the steady case and only a few more iterations in the unsteady case.

4.3.1 | Steady case

The results for the steady problem in both 2D domains with the three uniformly refined meshes are shown in Table 3. From these results it seems that AL is a very good preconditioner for the linear systems arising from the IgA discretizations of the steady incompressible flow. From the iteration count point of view, its convergence is very fast and robust. However, the ideal version is very expensive. An efficient approximate solver for the block F_γ is a real necessity for AL to be an efficient preconditioner.

TABLE 3 Number of GMRES iterations for the steady two-dimensional problems on three uniformly refined meshes

	Step			Profile				Step			Profile		
LSC	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**	**PCD**	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**
2-1, C^0	32	33	39	55	46	35	2-1, C^0	46	38	30	76	83	45
3-2, C^0	25	31	52	—	—	—	3-2, C^0	35	31	29	—	—	—
3-2, C^1	31	28	27	—	—	—	3-2, C^1	43	37	30	—	—	—
4-3, C^0	21	26	48	29	29	44	4-3, C^0	35	33	31	41	31	29
4-3, C^1	24	22	28	—	—	—	4-3, C^1	37	33	31	—	—	—
4-3, C^2	27	24	27	34	35	30	4-3, C^2	42	37	31	63	68	37
4-3, SG	32	33	36	31	39	41	4-3, SG	42	37	31	62	67	35
	Step			**Profile**				**Step**			**Profile**		
SIMPLE	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**	**SIMPLER**	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**
2-1, C^0	119	141	165	100	130	183	2-1, C^0	29	26	26	65	66	42
3-2, C^0	117	122	164	—	—	—	3-2, C^0	27	31	52	—	—	—
3-2, C^1	127	125	155	—	—	—	3-2, C^1	30	25	23	—	—	—
4-3, C^0	218	188	192	373	344	291	4-3, C^0	20	27	54	41	38	57
4-3, C^1	163	151	162	—	—	—	4-3, C^1	27	21	22	—	—	—
4-3, C^2	120	113	141	157	153	167	4-3, C^2	27	23	21	50	54	38
4-3, SG	117	114	143	147	150	170	4-3, SG	29	30	28	56	64	45
	Step			**Profile**				**Step**			**Profile**		
AL	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**	**MAL**	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**
2-1, C^0	8	7	6	7	6	6	2-1, C^0	52	66	77	78	135	172
3-2, C^0	7	5	5	—	—	—	3-2, C^0	87	104	111	—	—	—
3-2, C^1	8	6	5	—	—	—	3-2, C^1	60	79	90	—	—	—
4-3, C^0	6	5	5	5	5	4	4-3, C^0	116	136	140	249	324	324
4-3, C^1	7	5	5	—	—	—	4-3, C^1	99	114	121	—	—	—
4-3, C^2	12	6	5	7	5	5	4-3, C^2	65	90	100	107	181	256
4-3, SG	12	7	5	7	5	5	4-3, SG	62	84	92	105	175	245

SIMPLE and MAL give relatively slow convergence, which seems to be generally somewhat faster for higher order of continuity. Note that the convergence of MAL can be probably improved by tuning the parameter γ according to the particular problem and discretization.

LSC, PCD, SIMPLER, and MSIMPLER need a similar number of iterations to reach the required tolerance. LSC and the two SIMPLE-type preconditioners are robust with respect to uniform mesh refinement for the low-degree discretization, but their convergence slows down with uniform mesh refinement for higher degree C^0 discretizations (considered as analogues of higher degree FEM). However, their convergence does not depend on mesh refinement or even accelerates for discretizations of higher order of continuity. Figure 9 shows a comparison of the whole convergence curves of LSC and SIMPLER for various mesh refinement in the case of C^0 and C^2 isogeometric TH discretization of degrees 4-3 for the backward facing step.

There will be probably some slowdown of convergence if we further refine the mesh, but the dependence seems weaker for C^2 than in the C^0 case. Moreover, the SIMPLER preconditioner shows stagnation at the beginning of the iteration process, which prolongs with the mesh refinement. In the case of higher continuity, these stagnation periods are significantly shorter. We observe similar behavior for the blade profile geometry.

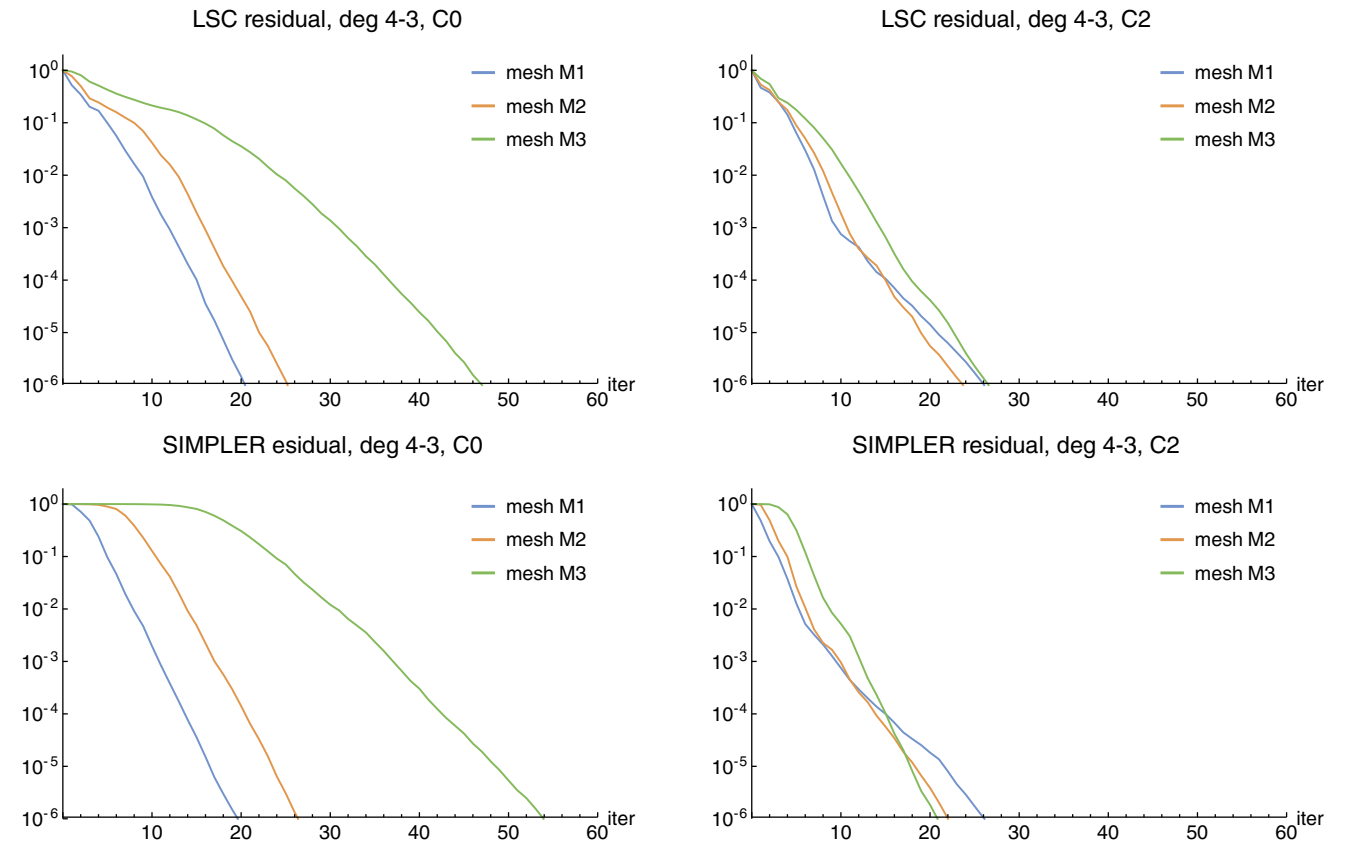

FIGURE 9 Backward facing step, steady: convergence of LSC and SIMPLER for various mesh sizes for degree 4-3 with C^0 (left) and C^2 (right) continuity [Colour figure can be viewed at wileyonlinelibrary.com]

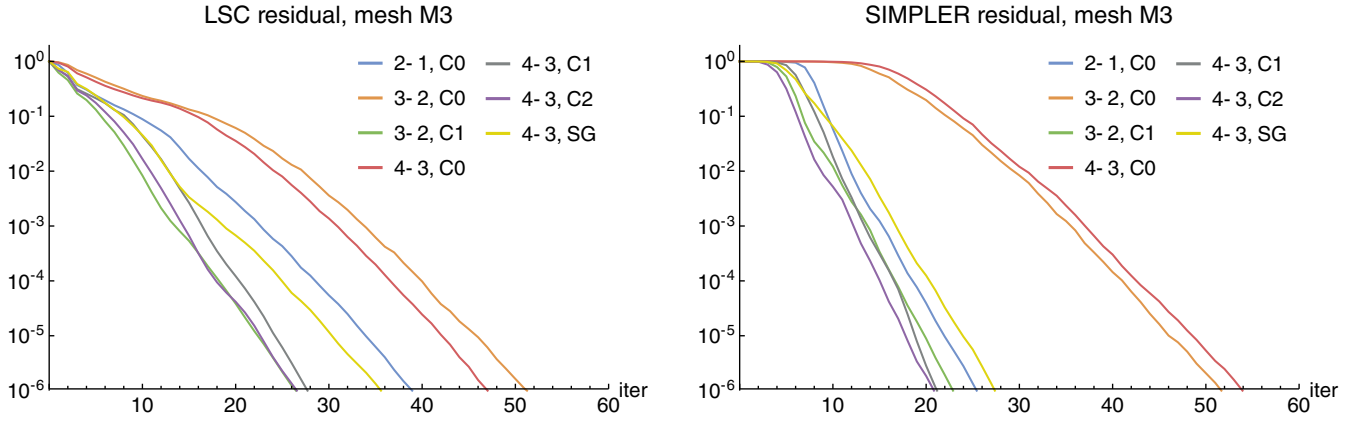

FIGURE 10 Backward facing step, steady: convergence of LSC and SIMPLER for various discretizations on the finest uniform mesh M3 [Colour figure can be viewed at wileyonlinelibrary.com]

If we consider the finest mesh M3, the convergence of LSC and SIMPLE-type preconditioners slows down for increasing the degree with C^0 continuity. On the contrary, it accelerates a bit if we increase both degree and continuity. See the comparison of the convergence curves of LSC and SIMPLER for the backward facing step in Figure 10.

We get the fastest convergence for the C^2 isogeometric TH element of degree 4-3. The convergence for the same degree SG element (i.e., C^3 continuity for velocity) is slightly slower in most cases. Again, we can make similar observations for the blade profile.

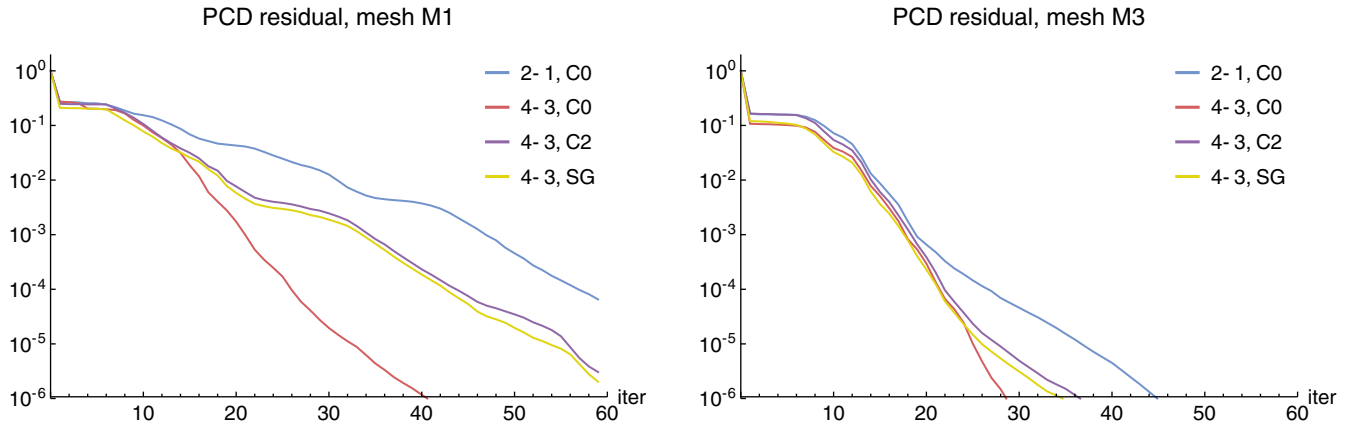

FIGURE 11 Blade profile, steady: convergence of PCD for various discretizations on the "uniform" mesh M1 (left) and M3 (right) [Colour figure can be viewed at wileyonlinelibrary.com]

TABLE 4 Number of GMRES iterations for the discretization of degree 4-3 with C^2 continuity on the finest "uniform" mesh M3 and two levels of its local refinement for the steady two-dimensional problems

Steady	Step			Profile		
4-3, C^2	M3	loc 1	loc 2	M3	loc 1	loc 2
LSC	27	34	61	30	45	84
PCD	31	30	29	37	37	37
SIMPLE	141	147	171	167	190	229
SIMPLER	21	27	39	38	51	68
AL	5	5	5	5	5	5
MAL	100	100	99	256	260	260

The convergence of the PCD preconditioner tends to improve with mesh refinement and mostly also for increasing degree of the discretization basis. However, it deteriorates a bit for increasing continuity of the basis, especially for coarser meshes. See the comparison of its convergence for various discretizations of the blade profile geometry with the mesh M1 and M3 in Figure 11.

We are also interested in the dependence of the convergence on the local refinement (i.e., the grid stretching). We have chosen the finest mesh M3 with the C^2 continuous discretization of degree 4-3 for demonstration. In Table 4, we present the numbers of iterations for this mesh and the two levels of its local refinement. Again, MSIMPLER is not included, since its behavior is similar to LSC. Based on these results, PCD, AL and MAL seem to be robust with respect to grid stretching. The convergence of LSC and the SIMPLE-type preconditioners slows down for stretched grids, the dependence seems to be weaker for SIMPLER. We generally observe similar behavior for other meshes and discretizations. Figure 12 shows a comparison of convergence curves of all preconditioners for the blade profile geometry with the "uniform" mesh M3 and the locally refined mesh denoted as "loc 2."

4.3.2 | Unsteady case

The results for the unsteady problems with the three uniformly refined meshes are summarized in Table 5. We can see a generally faster convergence than in the steady case for most preconditioners, which can be expected thanks to the presence of the velocity mass matrix in the block F of the system matrix. However, the numbers of iterations for the PCD preconditioner are similar as in the steady case. The convergence of AL is even slower, especially for the backward facing step, but it can be improved by tuning the parameter γ. Most of the preconditioners are robust with respect to mesh refinement, except for PCD, which shows some mesh dependence, and MAL in the case of the blade profile.

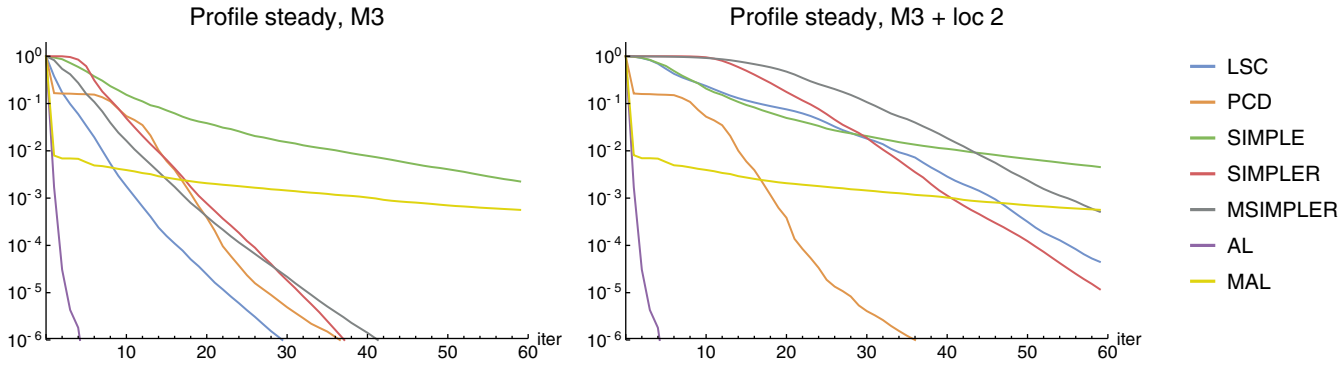

FIGURE 12 Blade profile, steady: convergence of various preconditioners for the "uniform" mesh M3 with C^2 continuous discretization of degree 4-3 (left) and the locally refined mesh (right) [Colour figure can be viewed at wileyonlinelibrary.com]

TABLE 5 Number of GMRES iterations for the unsteady two-dimensional problems on three uniformly refined meshes

	Step			Profile				Step			Profile		
LSC	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**	**PCD**	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**
2-1, C^0	6	6	5	4	4	6	2-1, C^0	18	25	35	18	25	34
3-2, C^0	9	9	9	—	—	—	3-2, C^0	24	31	40	—	—	—
3-2, C^1	7	6	6	—	—	—	3-2, C^1	19	25	33	—	—	—
4-3, C^0	12	11	10	13	14	17	4-3, C^0	32	40	39	45	55	39
4-3, C^1	10	9	8	—	—	—	4-3, C^1	26	35	39	—	—	—
4-3, C^2	7	7	6	6	6	8	4-3, C^2	22	27	31	24	32	36
4-3, SG	6	5	5	5	7	9	4-3, SG	23	27	31	24	32	36
	Step			**Profile**				**Step**			**Profile**		
SIMPLE	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**	**SIMPLER**	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**
2-1, C^0	8	7	6	8	8	9	2-1, C^0	7	6	5	6	6	7
3-2, C^0	24	21	14	—	—	—	3-2, C^0	10	10	9	—	—	—
3-2, C^1	13	11	9	—	—	—	3-2, C^1	8	7	6	—	—	—
4-3, C^0	66	55	35	75	74	59	4-3, C^0	14	12	11	19	20	22
4-3, C^1	37	32	23	—	—	—	4-3, C^1	12	11	9	—	—	—
4-3, C^2	18	15	12	17	15	14	4-3, C^2	9	7	7	8	8	8
4-3, SG	13	11	9	13	12	13	4-3, SG	7	6	6	7	8	11
	Step			**Profile**				**Step**			**Profile**		
AL	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**	**MAL**	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**
2-1, C^0	35	40	35	12	14	14	2-1, C^0	38	42	38	16	20	23
3-2, C^0	34	37	35	—	—	—	3-2, C^0	37	40	41	—	—	—
3-2, C^1	34	37	38	—	—	—	3-2, C^1	36	39	40	—	—	—
4-3, C^0	32	33	30	13	13	12	4-3, C^0	35	37	36	24	35	43
4-3, C^1	30	32	28	—	—	—	4-3, C^1	33	35	32	—	—	—
4-3, C^2	31	33	34	13	14	13	4-3, C^2	34	35	36	18	24	32
4-3, SG	31	33	33	13	14	13	4-3, SG	34	35	35	19	25	32

TABLE 6 Number of GMRES iterations for the discretization of degree 4-3 with C^2 continuity on the finest "uniform" mesh M3 and two levels of its local refinement for the unsteady two-dimensional problems

Unsteady	Step			Profile		
4-3, C^2	M3	loc 1	loc 2	M3	loc 1	loc 2
LSC	6	7	10	8	8	11
PCD	31	35	43	36	37	42
SIMPLE	12	12	20	14	13	21
SIMPLER	7	7	17	8	9	18
AL	34	34	34	13	13	13
MAL	36	42	45	32	37	44

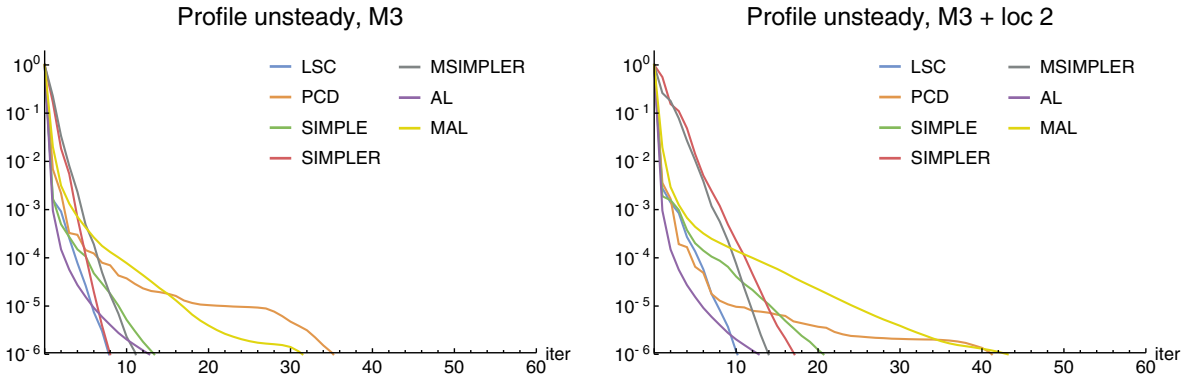

FIGURE 13 Blade profile, unsteady: convergence of various preconditioners for the "uniform" mesh M3 with C^2 continuous discretization of degree 4-3 (left) and the locally refined mesh (right) [Colour figure can be viewed at wileyonlinelibrary.com]

For all meshes in the unsteady case, we observe a similar phenomenon as for the finest mesh in the steady case: the number of iterations for LSC, SIMPLER and MSIMPLER slightly increases for increasing the degree with C^0 continuity, but stays almost constant if we increase both degree and continuity. The same also applies to PCD. The convergence of SIMPLE slows down for increasing degree in any case, but the dependence on the degree is much less significant for increasing continuity.

The results for stretched grids are presented in Table 6 for the unsteady case. The AL preconditioner is again independent of grid stretching and the convergence of all other preconditioners slows down only slightly for the locally refined meshes. Unlike the steady case, the PCD preconditioner is no longer independent of local refinement.

The comparison of convergence curves of all preconditioners for the blade profile geometry with the "uniform" mesh M3 and the locally refined mesh denoted as "loc 2" is displayed in Figure 13. For PCD, it can be seen that a rapid reduction of the residual norm happens in the first few iterations and then it almost stagnates.

Since the results for PCD may seem surprising in the unsteady case, we would like to emphasize again that these results were obtained for the original version of the preconditioner, where the discrete operators F_p and A_p are defined with Dirichlet boundary conditions at the inflow part of the boundary and Neumann conditions elsewhere. An improved choice of the boundary conditions can have a significant effect on the performance of the preconditioner. The implementation of other choices is a matter of future work on our solver.

In Table 7 we compare the convergence for the unsteady 2D backward facing step problem with three different values of Reynolds number, $Re = 100$, 1000, and 10,000, on the finest uniform mesh M3. Again, it seems that the higher continuity can have some benefits. Especially for PCD and SIMPLE there is a relatively significant increase of the number of iterations for increasing Re for the C^0 discretizations of higher degree. However, for increasing continuity, the dependence becomes much less significant. For LSC, SIMPLER, and MSIMPLER, the differences in iteration counts are small, but the case with the highest continuity always requires the least number of iterations. AL and MAL are mostly independent of the Reynolds number, although we set $\gamma = 1$ (independently of Re) for all cases.

TABLE 7 Number of GMRES iterations for the unsteady two-dimensional backward facing step for three values of *Re* on the mesh M3

	Re				*Re*				*Re*		
LSC	10^2	10^3	10^4	**PCD**	10^2	10^3	10^4	**SIMPLE**	10^2	10^3	10^4
2-1, C^0	5	7	7	2-1, C^0	35	39	39	2-1, C^0	6	7	7
3-2, C^0	9	12	13	3-2, C^0	40	49	50	3-2, C^0	14	23	26
3-2, C^1	6	8	8	3-2, C^1	33	38	38	3-2, C^1	9	10	10
4-3, C^0	10	12	13	4-3, C^0	39	66	80	4-3, C^0	35	97	136
4-3, C^1	8	11	12	4-3, C^1	39	57	63	4-3, C^1	23	39	45
4-3, C^2	6	8	8	4-3, C^2	31	38	40	4-3, C^2	12	15	16
4-3, SG	5	7	8	4-3, SG	31	39	42	4-3, SG	9	11	11
	Re				*Re*				*Re*		
SIMPLER	10^2	10^3	10^4	**AL**	10^2	10^3	10^4	**MAL**	10^2	10^3	10^4
2-1, C^0	5	6	6	2-1, C^0	35	44	44	2-1, C^0	38	47	47
3-2, C^0	9	13	14	3-2, C^0	35	37	37	3-2, C^0	41	46	46
3-2, C^1	6	8	8	3-2, C^1	38	38	38	3-2, C^1	40	41	41
4-3, C^0	11	14	17	4-3, C^0	30	32	32	4-3, C^0	36	50	53
4-3, C^1	9	14	15	4-3, C^1	28	32	32	4-3, C^1	32	41	42
4-3, C^2	7	8	9	4-3, C^2	34	34	34	4-3, C^2	36	37	37
4-3, SG	6	7	8	4-3, SG	33	33	33	4-3, SG	35	36	36

	DOFs		**LU time**	
Step	**4-3, C^0**	**4-3, C^2**	**4-3, C^0**	**4-3, C^2**
M1	10,997	3185	0.4	0.3
M2	44297	11,229	2.7	3.4
M3	177,809	42,005	43.6	24.5

TABLE 8 Number of degrees of freedom and CPU time of the LU decomposition for the meshes M1, M2 and M3 with 4-3, C^0 and 4-3, C^2 discretization of the two-dimensional backward facing step

4.3.3 | Computational time

Since the performance of our codes is not optimized, we only mention some CPU times and briefly comment on the comparison of the individual methods. We illustrate the situation for two discretizations of the 2D backward facing step of degree 4-3 with C^0 and C^2 continuity. In Table 8 we give the number of degrees of freedom for these discretizations with the three uniform meshes and the CPU time of the direct solution of the corresponding linear systems using the sequential sparse LU decomposition available from the Eigen library. Obviously, the direct solution of the linear systems resulting from C^2 continuous discretizations takes significantly longer than for systems of comparable size resulting from C^0 continuous discretizations. This is caused by higher density of the C^2 matrices, which is already documented in the literature.[4]

Figure 14 displays the computational times of the GMRES iterations needed to solve the linear systems obtained for the steady problem with the tolerance 10^{-6} (not including the preconditioner construction time, which involves the LU decomposition of the subsystems). The left and the middle panel show the computational time depending on the uniform mesh refinement level for C^0 and C^2 continuous discretization, respectively. Notice the different scale of the two graphs. The right panel shows these computational times for both discretizations depending on the number of degrees of freedom.

In the overall comparison, all tested preconditioners except MAL and SIMPLE seem comparably efficient for the C^2 discretization in this case, SIMPLER and PCD are slightly faster than the others. If we also take into account the local

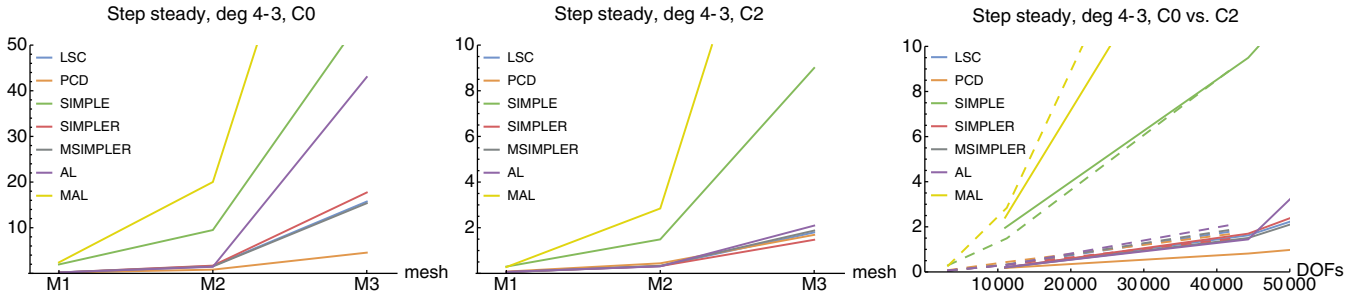

FIGURE 14 Two-dimensional backward facing step, steady. Left and center: Computational time (in seconds) of GMRES iterations with various preconditioners for the meshes M1, M2, M3 with C^0 (left) and C^2 (middle) continuous discretization of degree 4-3. Right: Comparison of these computational times for the C^0 (solid line) and C^2 (dashed line) continuous discretizations depending on the number of degrees of freedom [Colour figure can be viewed at wileyonlinelibrary.com]

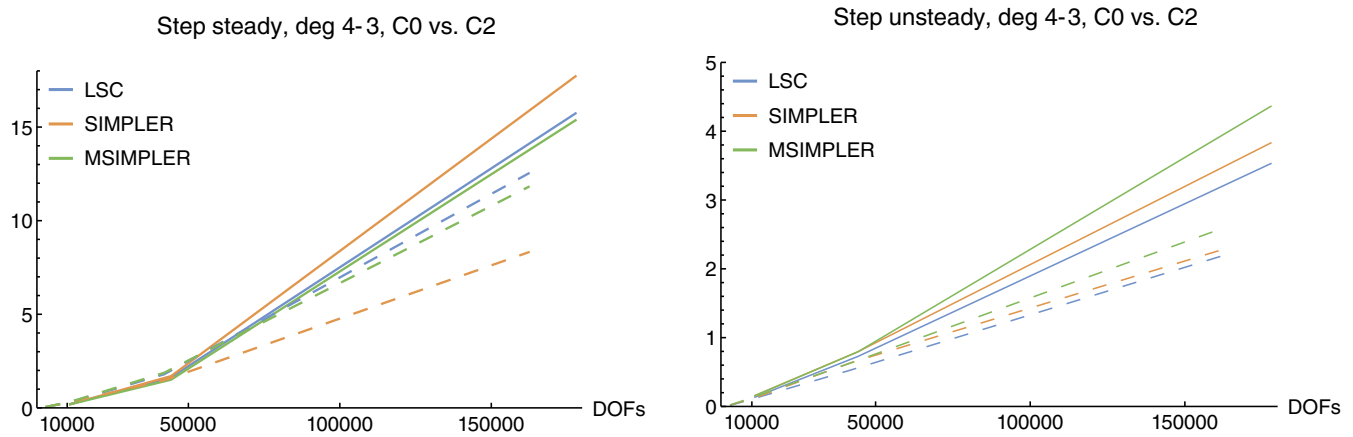

FIGURE 15 Two-dimensional backward facing step, steady (left) and unsteady (right): Computational time (in seconds) of GMRES iterations with selected preconditioners depending on the number of degrees of freedom, comparison of the C^0 (solid line) and C^2 (dashed line) continuous discretizations of degree 4-3 [Colour figure can be viewed at wileyonlinelibrary.com]

refinement, PCD, SIMPLER, and AL are the most efficient thanks to their robustness. For the unsteady case, all tested preconditioners except AL and MAL perform comparably.

The computational time of the GMRES iterations is comparable for the linear systems of similar size for C^0 and C^2 continuity for the tested cases. Moreover, if we compare the C^0 case with the mesh M3 and the C^2 case with a mesh obtained by additional uniform refinement of M3, which has a similar number of DOFs, LSC, SIMPLER, and MSIMPLER are even more efficient for the C^2 case thanks to the fact that the dependence of their convergence on the mesh refinement is stronger for C^0 discretizations. This is illustrated in Figure 15 for both steady and unsteady problem.

For a fair comparison of the efficiency of different preconditioners, it is important to take into account their setup time. In our case, for example, for the mesh M3 with the C^2 continuous discretization, the setup times are following: LSC and SIMPLE-type preconditioners around 3 s, PCD less than 2 s, AL 134 s and MAL 56 s. Thus, although the ideal version of AL may seem efficient if we measure only the time of the GMRES iterations, it is very expensive due to its construction, which involves the LU decomposition of the block F_γ. But let us emphasize again that we consider only the ideal versions of the preconditioners, which is impractical. The efficiency comparison may turn out differently if we use suitable approximate solvers for the preconditioner blocks.

4.4 | Results in 3D

For the 3D backward facing step problem, we present results for the three uniform meshes and a fixed $Re = 100$. The iteration counts needed to solve the linear systems obtained from discretization of the steady and unsteady problems

TABLE 9 Number of GMRES iterations for the steady three-dimensional backward facing step

LSC	M1	M2	M3	PCD	M1	M2	M3	SIMPLE	M1	M2	M3
2-1, C^0	23	28	34	2-1, C^0	40	37	33	2-1, C^0	63	93	128
3-2, C^0	25	23	32	3-2, C^0	38	37	35	3-2, C^0	212	184	145
3-2, C^1	23	26	29	3-2, C^1	38	36	33	3-2, C^1	117	144	124
4-3, C^0	31	22	x	4-3, C^0	46	46	42	4-3, C^0	555	572	456
4-3, C^1	37	27	25	4-3, C^1	42	42	40	4-3, C^1	378	357	310
4-3, C^2	25	23	26	4-3, C^2	40	37	37	4-3, C^2	208	188	137
4-3, SG	22	28	35	4-3, SG	37	37	37	4-3, SG	152	155	119
SIMPLER	**M1**	**M2**	**M3**	**AL**	**M1**	**M2**	**M3**	**MAL**	**M1**	**M2**	**M3**
2-1, C^0	21	24	26	2-1, C^0	8	7	5	2-1, C^0	41	84	128
3-2, C^0	23	23	32	3-2, C^0	9	5	x	3-2, C^0	104	178	x
3-2, C^1	23	25	21	3-2, C^1	8	7	x	3-2, C^1	50	101	x
4-3, C^0	30	21	31	4-3, C^0	8	x	x	4-3, C^0	185	x	x
4-3, C^1	36	28	21	4-3, C^1	8	5	x	4-3, C^1	120	210	x
4-3, C^2	24	22	18	4-3, C^2	13	8	x	4-3, C^2	64	117	x
4-3, SG	22	23	24	4-3, SG	12	9	x	4-3, SG	61	111	x

TABLE 10 Number of GMRES iterations for the unsteady three-dimensional backward facing step

LSC	M1	M2	M3	PCD	M1	M2	M3	SIMPLE	M1	M2	M3
2-1, C^0	7	7	6	2-1, C^0	18	30	50	2-1, C^0	13	12	10
3-2, C^0	10	10	10	3-2, C^0	27	47	71	3-2, C^0	62	62	51
3-2, C^1	9	8	7	3-2, C^1	23	35	56	3-2, C^1	27	24	20
4-3, C^0	15	14	x	4-3, C^0	55	132	294	4-3, C^0	211	237	194
4-3, C^1	13	12	10	4-3, C^1	44	85	212	4-3, C^1	120	126	111
4-3, C^2	11	8	7	4-3, C^2	31	52	92	4-3, C^2	56	43	35
4-3, SG	8	6	6	4-3, SG	33	54	91	4-3, SG	31	24	20
SIMPLER	**M1**	**M2**	**M3**	**AL**	**M1**	**M2**	**M3**	**MAL**	**M1**	**M2**	**M3**
2-1, C^0	8	7	6	2-1, C^0	20	24	26	2-1, C^0	23	27	29
3-2, C^0	13	12	11	3-2, C^0	18	20	x	3-2, C^0	22	23	x
3-2, C^1	10	9	8	3-2, C^1	18	20	x	3-2, C^1	21	23	x
4-3, C^0	19	16	15	4-3, C^0	17	x	x	4-3, C^0	21	x	x
4-3, C^1	17	15	13	4-3, C^1	17	17	x	4-3, C^1	21	20	x
4-3, C^2	13	10	8	4-3, C^2	18	17	x	4-3, C^2	21	20	x
4-3, SG	10	8	6	4-3, SG	18	17	x	4-3, SG	21	20	x

are summarized in Tables 9 and 10, respectively. In some cases, the computation failed due to insufficient memory, which is indicated by ”x” in the tables. Again, we do not present the results for MSIMPLER for brevity; it requires one or two iterations less than LSC in the steady case and converges very similarly to SIMPLER in the unsteady case for all discretizations.

The overall convergence behavior with respect to the discretization degree and continuity is similar as for the unsteady problem in 2D. For all tested preconditioners except AL (and MAL in the unsteady case), the number of iterations slightly increases for increasing the degree with C^0 continuity, but stays almost constant, or at least the dependence is much less

significant, if we increase both degree and continuity. For the steady problem with finer meshes, the convergence of LSC, SIMPLER and MSIMPLER even accelerates a bit for increasing degree and continuity for the discretizations with the isogeometric TH element.

In the steady case, the AL preconditioner is robust and requires the least number of iterations of all preconditioners, similarly to the 2D test problems. LSC, PCD, SIMPLER, and MSIMPLER require a similar number of iterations and their convergence is quite robust with respect to the discretization degree and continuity.

In the unsteady case, LSC, SIMPLER ,and MSIMPLER appear to be the most effective from the iteration count point of view, but note that the convergence of AL and MAL can be influenced by tuning the parameter γ. All mentioned preconditioners are relatively robust with respect to both mesh refinement and discretization degree and continuity.

5 | CONCLUSION

In this paper, several block preconditioners for the linearized Navier–Stokes equations were compared for the isogeometric discretizations of two model problems in two dimensions and one in three dimensions. The selected preconditioning techniques are known from the literature mainly in the connection with (usually low degree) FEM discretizations. For IgA, discretization bases of high degree and high order of interelement continuity are common, where the latter property is unique for IgA. The comparison focused mainly on the performance of the ideal versions of these preconditioners for various polynomial degrees and orders of continuity of the IgA discretization bases.

Our comparison of the individual preconditioners suggests that thanks to their robustness and computational costs, PCD and SIMPLER are a good choice in the steady case. In the unsteady case, LSC and MSIMPLER are also very good. AL is generally a very effective preconditioner which is robust with respect to the problem parameters and also to the discretization degree and continuity. However, the ideal version is very expensive and thus a fast approximate solver for the augmented block F_γ is necessary.

Based on our experiments, we can conclude that high interelement continuity of the discretization bases does not spoil the convergence properties of GMRES with the tested preconditioners, moreover, the opposite is true in many cases. For example, LSC, SIMPLER and MSIMPLER seem to be less sensitive to uniform mesh refinement for discretizations of higher continuity in the 2D steady case. We observe a similar behavior, especially for PCD and SIMPLE, regarding the dependence on the Reynolds number for a 2D unsteady problem. In the unsteady case in 2D and also for the 3D test problem in general, all tested preconditioners except AL and MAL converge faster for higher continuity discretizations than for C^0 discretization of the same degree. Thus, the potential efficiency of IgA will probably not be spoiled by the use of iterative solvers with the block preconditioners.

The practical efficiency of the iterative solution of the IgA-based linear systems using block preconditioners will depend on the availability of approximate solvers for the subsystems suitable for IgA discretizations. This can be challenging, especially in the case of the block F_γ in the AL preconditioner. Incorporating and testing suitable approximate inner solvers is a topic for future work.

ACKNOWLEDGEMENTS

This work was supported by the Czech Science Foundation (GAČR) grant No. 19-04006S.

DATA AVAILABILITY STATEMENT

The data that supports the findings of this study are available in this article.

ORCID

Hana Horníková https://orcid.org/0000-0002-0141-1907

REFERENCES

1. Hughes T, Cottrel J, Bazilevs Y. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Comput Methods Appl Mech Eng*. 2005;194:4135-4195.
2. Cottrell J, Hughes T, Bazilevs Y. *Isogeometric Analysis: Toward Integration of CAD and FEA*. John Wiley & Sons Ltd: Hoboken, NJ; 2009.
3. Collier N, Dalcin L, Pardo D, Calo VM. The cost of continuity: performance of iterative solvers on isogeometric finite elements. *SIAM J Sci Comput*. 2013;35(2):767-784.

4. Collier N, Pardo D, Dalcin L, Paszynski M, Calo VM. The cost of continuity: a study of the performance of isogeometric finite elements using direct solvers. *Comput Methods Appl Mech Eng*. 2012;213–216:353-361.
5. Benzi M, Golub G, Liesen J. Numerical solution of saddle point problems. *Acta Numerica*. 2005;14:1-137.
6. Segal A, Ur Rehman M, Vuik C. Preconditioners for incompressible Navier–Stokes solvers. *Numer Math Theor Meth Appl*. 2010;3(3):245-275.
7. ur Rehman M, Vuik C, Segal AA. comparison of preconditioners for incompressible Navier–Stokes solvers. *Int J Numer Methods Fluids*. 2008;57:1731-1751.
8. Kay D, Loghin D, Wathen A. A preconditioner for the steady-state Navier–Stokes equations. *SIAM J Sci Comput*. 2002;24(1):237-256.
9. Silvester D, Elman H, Kay D, Wathen A. Efficient preconditioning of the linearised Navier–Stokes equations for incompressible flow. *J Comput Appl Math*. 2001;128:261-279.
10. Elman H, Howle VE, Shadid J, Shuttleworth R, Tuminaro R. Block preconditioners based on approximate commutators. *SIAM J Sci Comput*. 2006;27(5):1651-1668.
11. Elman H. Preconditioning for the steady-state Navier–Stokes equations with low viscosity. *SIAM J Sci Comput*. 1999;20(4):1299-1316.
12. Benzi M, Olshanskii MA. An augmented Lagrangian-based approach to the Oseen problem. *SIAM J Sci Comput*. 2006;28(6):2095-2113.
13. Vuik C, Saghir A. The Krylov accelerated SIMPLE(R) method for incompressible flow. Technical Report; Delft University of Technology; 2002.
14. Vuik C, Saghir A, Boerstoel GP. The Krylov accelerated SIMPLE(R) method for flow problems in industrial furnaces. *Int J Numer Methods Fluids*. 2000;33:1027-1040.
15. Klaij CM, Vuik C. SIMPLE-type preconditioners for cell-centered, colocated finite volume discretization of incompressible Reynolds-averaged Navier–Stokes equations. *Int J Numer Methods Fluids*. 2013;17:830-849.
16. Côrtes AMA, Dalcin L, Sarmiento A, Collier N, Calo VM. A scalable block-preconditioning strategy for divergence-conforming B-spline discretizations of the Stokes problem. *Comput Methods Appl Mech Eng*. 2017;316:839-858.
17. Horníková H, Vuik C. Preconditioning for linear systems arising from IgA discretized incompressible Navier–Stokes equations. Isogeometric Analysis and Applications. Lecture Notes in Computational Science and Engineering; 2018.
18. Fouchet-Incaux J. Artificial boundaries and formulations for the incompressible Navier–Stokes equations. applications to air and blood flows. *SeMA J*. 2014;64:1-40.
19. Heywood JG, Rannacher R, Turek S. Artificial boundaries and flux and pressure conditions for the incompressible Navier–Stokes equations. *Int J Numer Methods Fluids*. 1996;22:325-352.
20. Brezzi F, Fortin M. *Mixed and Hybrid Finite Element Methods*. New York, NY: Springer-Verlag; 1991.
21. de Boor CA. *Practical Guide to Splines. Vol. 27 of Applied Mathematical Sciences*. New York, NY: Springer; 2001 Revised Edition.
22. Bressan A, Sangalli G. Isogeometric discretizations of the Stokes problem: stability analysis by the macroelement technique. *IMA J Numer Anal*. 2012;33(2):629-651.
23. Nielsen PN, Gersborg AR, Gravesen J, Pedersen NL. Discretizations in isogeometric analysis of Navier–Stokes flow. *Comput Methods Appl Mech Eng*. 2011;200(45):3242-3253.
24. Murphy MF, Golub GH, Wathen AJ. A note on preconditioning for indefinite linear systems. *SIAM J Sci Comput*. 2000;21(6):1969-1972.
25. Elman H, Silvester D, WA. *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics*. 2nd ed. Oxford, UK: Oxford University Press; 2014.
26. Donatelli M, Garoni C, Manni C, Serra-Capizzano S, Speleers H. Symbol-based multigrid methods for Galerkin B-spline isogeometric analysis. *SIAM J Numer Anal*. 2017;55(1):31-62.
27. Hofreither C, Takacs S. Robust multigrid for isogeometric analysis based on stable splittings of spline spaces. *SIAM J Numer Anal*. 2017;55(4):2004-2024.
28. Hofer C, Takacs S. A parallel multigrid solver for multi-patch isogeometric analysis. *Advanced Finite Element Methods with Applications. FEM 2017*. Lecture Notes in Computational Science and Engineering Vol 128. Cham: Springer; 2019:205-219.
29. de la Riva AP, Rodrigo C, Gaspar FJ. A robust multigrid solver for isogeometric analysis based on multiplicative Schwarz smoothers. *SIAM J Sci Comput* 2019; 41(5): S321-S345.
30. Tielen R, Möller M, Göddeke D, Vuik C. p-multigrid methods and their comparison to h-multigrid methods within Isogeometric Analysis. *Comput Methods Appl Mech Eng*. 2020;372:113347
31. Buffa A, Harbrecht H, Kunoth A, Sangalli G. BPX-preconditioning for isogeometric analysis. *Comput Methods Appl Mech Eng*. 2013;265:63-70.
32. Sangalli G, Tani M. Isogeometric preconditioners based on fast solvers for the Sylvester equation. *SIAM J Sci Comput*. 2016;38(6):A3644-A3671.
33. Loghin D. Analysis of preconditioned picard iterations for the Navier–Stokes equations; 2001.
34. Elman H, Tuminaro R. Boundary conditions in approximate commutator preconditioners for the Navier–Stokes equations. *Electron Trans Numer Anal*. 2009;35(27):257-280.
35. Farrell PE, Mitchell L, Wechsung F. An augmented lagrangian preconditioner for the 3D stationary incompressible navier–stokes equations at high Reynolds number. *SIAM J Sci Comput*. 2019;41(5):A3073-A3096.
36. Benzi M, Olshanskii MA, Wang Z. Modified augmented Lagrangian preconditioners for the incompressible Navier–Stokes equations. *Int J Numer Methods Fluids*. 2011;66(4):486-508.
37. Patankar SV, Spalding DB. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *Int J Heat Mass Transf*. 1972;15:1787-1805.

38. Wesseling P. *Principles of Computational Fluid Dynamics*. Springer Series in Computational Mathematics. Vol 29. New York, NY: Springer; 2001.
39. ur Rehman M, Vuik C, Segal G. SIMPLE-type preconditioners for the Oseen problem. *Int J Numer Methods Fluids*. 2009;61(4):432-452.
40. Mantzaflaris A. G+Smo (Geometry plus simulation modules) v0.8.1; 2018. http://github.com/gismo. Accessed 10 Jan 2020.
41. Guennebaud G, Jacob B. Eigen v3; 2010. http://eigen.tuxfamily.org. Accessed 10 Jan 2020.

How to cite this article: Horníková H, Vuik C, Egermaier J. A comparison of block preconditioners for isogeometric analysis discretizations of the incompressible Navier–Stokes equations. *Int J Numer Meth Fluids*. 2021;93:1788–1815. https://doi.org/10.1002/fld.4952