Journal of Scientific Computing (2021) 89:20
https://doi.org/10.1007/s10915-021-01567-z

Fast Parallel Solver for the Space-time IgA-DG Discretization of the Diffusion Equation

Pietro Benedusi[1] · Paola Ferrari[2] · Carlo Garoni[3] · Rolf Krause[1] · Stefano Serra-Capizzano[4,5]

Received: 3 October 2020 / Revised: 3 July 2021 / Accepted: 7 July 2021 / Published online: 1 September 2021

Abstract
We consider the space-time discretization of the diffusion equation, using an isogeometric analysis (IgA) approximation in space and a discontinuous Galerkin (DG) approximation in time. Drawing inspiration from a former spectral analysis, we propose for the resulting space-time linear system a multigrid preconditioned GMRES method, which combines a preconditioned GMRES with a standard multigrid acting only in space. The performance of the proposed solver is illustrated through numerical experiments, which show its competitiveness in terms of iteration count, run-time and parallel scaling.

Keywords Isogeometric analysis · Discontinuous Galerkin · Preconditioned GMRES · Multigrid · Parallel solver · Spectral distribution · Diffusion equation

Mathematics Subject Classification 65M60 · 65F08 · 65M55 · 65Y05 · 47B06 · 35Q79

✉ Carlo Garoni
garoni@mat.uniroma2.it

Pietro Benedusi
pietro.benedusi@usi.ch

Paola Ferrari
pferrari@uninsubria.it

Rolf Krause
rolf.krause@usi.ch

Stefano Serra-Capizzano
stefano.serrac@uninsubria.it; stefano.serra@it.uu.se

1 Euler Institute, University of Italian Switzerland (USI), Lugano, Switzerland

2 Department of Science and High Technology, University of Insubria, Como, Italy

3 Department of Mathematics, University of Rome Tor Vergata, Rome, Italy

4 Department of Humanities and Innovation, University of Insubria, Como, Italy

5 Division of Scientific Computing, Department of Information Technology, Uppsala University, Uppsala, Sweden

1 Introduction

In recent years, with ever increasing computational capacities, space-time methods have received fast growing attention from the scientific community. Space-time approximations of dynamic problems, in contrast to standard time-stepping techniques, enable full space-time parallelism on modern massively parallel architectures [27]. Moreover, they can naturally deal with moving domains [38,57–59,63] and allow for space-time adaptivity [1,24,28,39,47,49,61]. The main idea of space-time formulations is to consider the temporal dimension as an additional spatial one and assemble a large space-time system to be solved in parallel as in [25]. Space-time methods have been used in combination with various numerical techniques, including finite differences [2,11,35], finite elements [4,26,37,40], isogeometric analysis [34,41], and discontinuous Galerkin methods [1,16,32,37,38,48,57,63]. Moreover, they have been considered for a variety of applications, such as mechanics [15], fluid dynamics [11,38,54], fluid-structure interaction [60], and many others. When dealing with space-time finite elements, the time direction needs special care. To ensure that the information flows in the positive time direction, a particular choice of the basis in time is often used. The discontinuous Galerkin formulation with an "upwind" flow is a common choice in this context; see, for example, [38,51,57,62].

Specialized parallel solvers have been recently developed for the large linear systems arising from space-time discretizations. We mention in particular the space-time parallel multigrid proposed by Gander and Neumüller [29], the parallel preconditioners for space-time isogeometric analysis proposed by Hofer et al. [34], the fast diagonalization techniques proposed by Langer and Zank [42] and Loli et al. [44], and the parallel proposal by McDonald and Wathen [46]. We also refer the reader to [56] for a recent review on space-time methods for parabolic evolution equations, and to [55] for algebraic multigrid methods.

In the present paper, we focus on the diffusion equation

$$\begin{cases} \partial_t u(t,\mathbf{x}) - \nabla \cdot K(\mathbf{x}) \nabla u(t,\mathbf{x}) = f(t,\mathbf{x}), & (t,\mathbf{x}) \in (0,\mathrm{T}) \times (0,1)^d, \\ u(t,\mathbf{x}) = 0, & (t,\mathbf{x}) \in (0,\mathrm{T}) \times \partial((0,1)^d), \\ u(t,\mathbf{x}) = 0, & (t,\mathbf{x}) \in \{0\} \times (0,1)^d, \end{cases} \tag{1.1}$$

where $K(\mathbf{x}) \in \mathbb{R}^{d\times d}$ is the matrix of diffusion coefficients and $f(t,\mathbf{x})$ is a source term. It is assumed that $K(\mathbf{x})$ is symmetric positive definite at every point $\mathbf{x} \in (0,1)^d$ and each component of $K(\mathbf{x})$ is a continuous bounded function on $(0,1)^d$. We impose homogeneous Dirichlet initial/boundary conditions both for simplicity and because the inhomogeneous case reduces to the homogeneous case by considering a lifting of the boundary data [50]. We consider for (1.1) the same space-time approximation as in [10], involving a $\boldsymbol{p}$-degree $C^{\boldsymbol{k}}$ isogeometric analysis (IgA) discretization in space and a q-degree discontinuous Galerkin (DG) discretization in time. Here, $\boldsymbol{p} = (p_1,\ldots,p_d)$ and $\boldsymbol{k} = (k_1,\ldots,k_d)$, where $\mathbf{0} \le \boldsymbol{k} \le \boldsymbol{p} - \mathbf{1}$ (i.e., $0 \le k_i \le p_i - 1$ for all $i = 1,\ldots,d$) and the parameters p_i and k_i represent, respectively, the polynomial degree and the smoothness of the IgA basis functions in direction x_i.

The overall discretization process leads to solving a large space-time linear system. We propose a fast solver for this system in the case of maximal smoothness $\boldsymbol{k} = \boldsymbol{p} - \mathbf{1}$, i.e., the case corresponding to the classical IgA paradigm [3,9,17,36]. The solver is a preconditioned GMRES (PGMRES) method whose preconditioner $\tilde{P}$ is obtained as an approximation of another preconditioner P inspired by the spectral analysis carried out in [10]. Informally speaking, the preconditioner $\tilde{P}$ is a standard multigrid, which is applied only in space and not

in time, and which involves, at all levels, a single symmetric Gauss–Seidel post-smoothing step and standard bisection for the interpolation and restriction operators (following the Galerkin assembly). The proposed solver is then a multigrid preconditioned GMRES (MG-GMRES). Its performance is illustrated through numerical experiments and turns out to be satisfactory in terms of iteration count and run-time. In addition, the solver is suited for parallel computation as it shows remarkable scaling properties with respect to the number of cores. Comparisons with other benchmark solvers are also presented and reveal the actual competitiveness of our proposal.

The paper is organized as follows. In Sect. 2, we briefly recall the space-time IgA-DG discretization of (1.1) and we report the main result of [10] concerning the spectral distribution of the associated discretization matrix C. In Sect. 3, we present a PGMRES method for the matrix C, which is the root from which the proposed solver originated. In Sect. 4, we describe the proposed solver. In Sect. 5, we describe its parallel version. In Sect. 6, we illustrate its performance in terms of iteration count, run-time and scaling. In Sect. 7, we test it on a generalization of problem (1.1) where $(0,1)^d$ is replaced by a non-rectangular domain and the considered IgA discretization involves a non-trivial geometry. In Sect. 8, we draw conclusions. In order to keep this paper as concise as possible, we borrow notation and terminology from [10]. It is therefore recommended that the reader takes a look at Sects. 1 and 2 of [10].

2 Space-time IgA-DG Discretization of the Diffusion Equation

Let $N \in \mathbb{N}$ and $\boldsymbol{n} = (n_1, \ldots, n_d) \in \mathbb{N}^d$, and define the following uniform partitions in time and space:

$$\begin{aligned} &t_i = i\,\Delta t, && i = 0, \ldots, N,\\ &\mathbf{x}_{\boldsymbol{i}} = \boldsymbol{i}\,\Delta\mathbf{x} = (i_1\Delta x_1, \ldots, i_d\Delta x_d), && \boldsymbol{i} = \mathbf{0}, \ldots, \boldsymbol{n}, \end{aligned}$$

where $\Delta t = \mathrm{T}/N$ and $\Delta\mathbf{x} = (\Delta x_1, \ldots, \Delta x_2) = (1/n_1, \ldots, 1/n_d)$. We consider for the differential problem (1.1) the same space-time discretization as in [10], i.e., we use a $\boldsymbol{p}$-degree $C^{\boldsymbol{k}}$ IgA approximation in space based on the uniform mesh $\{\mathbf{x}_{\boldsymbol{i}},\ \boldsymbol{i} = \mathbf{0}, \ldots, \boldsymbol{n}\}$ and a q-degree DG approximation in time based on the uniform mesh $\{t_i,\ i = 0, \ldots, N\}$. Here, $\boldsymbol{p} = (p_1, \ldots, p_d)$ and $\boldsymbol{k} = (k_1, \ldots, k_d)$ are multi-indices, with p_i and $0 \le k_i \le p_i - 1$ representing, respectively, the polynomial degree and the smoothness of the IgA basis functions in direction x_i. As explained in [10, Sect. 3], the overall discretization process leads to a linear system

$$C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)\mathbf{u} = \mathbf{f}, \tag{2.1}$$

where:

– $C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$ is the $N \times N$ block matrix given by

$$C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K) = \begin{bmatrix} A_{\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K) & & & \\ B_{\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]} & A_{\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K) & & \\ & \ddots & \ddots & \\ & & B_{\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]} & A_{\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K) \end{bmatrix}; \tag{2.2}$$

– the blocks $A_{\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$ and $B_{\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}$ are $(q+1)\bar{n} \times (q+1)\bar{n}$ matrices given by

$$A_{\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K) = K_{[q]} \otimes M_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]} + \frac{\Delta t}{2} M_{[q]} \otimes K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K), \tag{2.3}$$

$$B_{\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]} = -J_{[q]} \otimes M_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}, \tag{2.4}$$

where $\bar{n} = \prod_{i=1}^{d}(n_i(p_i - k_i) + k_i - 1)$ is the number of degrees of freedom (DoFs) in space (the total number of DoFs is equal to the size $N(q+1)\bar{n}$ of the matrix $C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$); each block row in the block partition of $C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$ given by (2.2) is referred to as a time slab;

– $M_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}$ and $K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K)$ are the $\bar{n} \times \bar{n}$ mass and stiffness matrices in space, which are given by

$$M_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]} = \left[\int_{[0,1]^d} B_{\boldsymbol{j}+\boldsymbol{1},[\boldsymbol{p},\boldsymbol{k}]}(\mathbf{x}) B_{\boldsymbol{i}+\boldsymbol{1},[\boldsymbol{p},\boldsymbol{k}]}(\mathbf{x}) \mathrm{d}\mathbf{x}\right]_{\boldsymbol{i},\boldsymbol{j}=\boldsymbol{1}}^{\boldsymbol{n}(\boldsymbol{p}-\boldsymbol{k})+\boldsymbol{k}-\boldsymbol{1}}, \tag{2.5}$$

$$K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K) = \left[\int_{[0,1]^d} \left[K(\mathbf{x}) \nabla B_{\boldsymbol{j}+\boldsymbol{1},[\boldsymbol{p},\boldsymbol{k}]}(\mathbf{x})\right] \cdot \nabla B_{\boldsymbol{i}+\boldsymbol{1},[\boldsymbol{p},\boldsymbol{k}]}(\mathbf{x}) \mathrm{d}\mathbf{x}\right]_{\boldsymbol{i},\boldsymbol{j}=\boldsymbol{1}}^{\boldsymbol{n}(\boldsymbol{p}-\boldsymbol{k})+\boldsymbol{k}-\boldsymbol{1}}, \tag{2.6}$$

where $B_{\boldsymbol{1},[\boldsymbol{p},\boldsymbol{k}]}, \ldots, B_{\boldsymbol{n}(\boldsymbol{p}-\boldsymbol{k})+\boldsymbol{k}+\boldsymbol{1},[\boldsymbol{p},\boldsymbol{k}]}$ are the tensor-product B-splines defined by

$$B_{\boldsymbol{i},[\boldsymbol{p},\boldsymbol{k}]}(\mathbf{x}) = \prod_{r=1}^{d} B_{i_r,[p_r,k_r]}(x_r), \quad \boldsymbol{i} = \boldsymbol{1}, \ldots, \boldsymbol{n}(\boldsymbol{p}-\boldsymbol{k}) + \boldsymbol{k} + \boldsymbol{1},$$

and $B_{1,[p_r,k_r]}, \ldots, B_{n_r(p_r-k_r)+k_r+1,[p_r,k_r]}$ are the B-splines of degree p_r and smoothness C^{k_r} defined on the knot sequence

$$\left\{\underbrace{0,\ldots,0}_{p_r+1}, \underbrace{\frac{1}{n_r},\ldots,\frac{1}{n_r}}_{p_r-k_r}, \underbrace{\frac{2}{n_r},\ldots,\frac{2}{n_r}}_{p_r-k_r}, \ldots, \underbrace{\frac{n_r-1}{n_r},\ldots,\frac{n_r-1}{n_r}}_{p_r-k_r}, \underbrace{1,\ldots,1}_{p_r+1}\right\}.$$

– $M_{[q]}$, $K_{[q]}$, $J_{[q]}$ are the $(q+1) \times (q+1)$ blocks given by

$$M_{[q]} = \left[\int_{-1}^{1} \ell_{j,[q]}(\tau) \ell_{i,[q]}(\tau) \mathrm{d}\tau\right]_{i,j=1}^{q+1}, \tag{2.7}$$

$$K_{[q]} = \left[\ell_{j,[q]}(1)\ell_{i,[q]}(1) - \int_{-1}^{1} \ell_{j,[q]}(\tau) \ell'_{i,[q]}(\tau) \mathrm{d}\tau\right]_{i,j=1}^{q+1}, \tag{2.8}$$

$$J_{[q]} = \left[\ell_{j,[q]}(1)\ell_{i,[q]}(-1)\right]_{i,j=1}^{q+1}, \tag{2.9}$$

where $\{\ell_{1,[q]}, \ldots, \ell_{q+1,[q]}\}$ is a fixed basis for the space of polynomials of degree $\leq q$. In the context of (nodal) DG methods [33], $\ell_{1,[q]}, \ldots, \ell_{q+1,[q]}$ are often chosen as the Lagrange polynomials associated with $q+1$ fixed points $\{\tau_1, \ldots, \tau_{q+1}\} \subseteq [-1, 1]$, such as, for example, the Gauss–Lobatto or the right Gauss–Radau nodes in $[-1, 1]$.

The solution of system (2.1) yields the approximate solution of problem (1.1); see [10] for details. The main result of [10] is reported in Theorem 2.1 below; see also [8, Sect. 6.2] for a more recent and lucid proof. Before stating Theorem 2.1, let us recall the notion of spectral distribution for a given sequence of matrices. In what follows, we say that a matrix-valued function $\mathbf{f} : D \to \mathbb{C}^{s\times s}$, defined on a measurable set $D \subseteq \mathbb{R}^{\ell}$, is measurable if its components $f_{ij} : D \to \mathbb{C}$, $i, j = 1, \ldots, s$, are (Lebesgue) measurable.

Definition 2.1 Let $\{X_m\}_m$ be a sequence of matrices, with X_m of size d_m tending to infinity, and let $\mathbf{f} : D \to \mathbb{C}^{s\times s}$ be a measurable matrix-valued function defined on a set $D \subset \mathbb{R}^\ell$ with $0 < \text{measure}(D) < \infty$. We say that $\{X_m\}_m$ has a (asymptotic) spectral distribution described by $\mathbf{f}$, and we write $\{X_m\}_m \sim_\lambda \mathbf{f}$, if

$$\lim_{m\to\infty} \frac{1}{d_m} \sum_{j=1}^{d_m} F(\lambda_j(X_m)) = \frac{1}{\text{measure}(D)} \int_D \frac{\sum_{i=1}^{s} F(\lambda_i(\mathbf{f}(\mathbf{y})))}{s} \mathrm{d}\mathbf{y}$$

for all continuous functions $F : \mathbb{C} \to \mathbb{C}$ with compact support. In this case, $\mathbf{f}$ is called the spectral symbol of $\{X_m\}_m$.

Remark 2.1 The informal meaning behind Definition 2.1 is the following: assuming that $\mathbf{f}$ possesses s Riemann-integrable eigenvalue functions $\lambda_i(\mathbf{f}(\mathbf{y}))$, $i = 1, \ldots, s$, the eigenvalues of X_m, except possibly for $o(d_m)$ outliers, can be subdivided into s different subsets of approximately the same cardinality; and the eigenvalues belonging to the ith subset are approximately equal to the samples of the ith eigenvalue function $\lambda_i(\mathbf{f}(\mathbf{y}))$ over a uniform grid in the domain D. For instance, if $\ell = 1$, $d_m = ms$, and $D = [a, b]$, then, assuming we have no outliers, the eigenvalues of X_m are approximately equal to

$$\lambda_i\Big(\mathbf{f}\Big(a + j\frac{b-a}{m}\Big)\Big), \quad j = 1, \ldots, m, \quad i = 1, \ldots, s,$$

for m large enough; similarly, if $\ell = 2$, $d_m = m^2 s$, and $D = [a_1, b_1] \times [a_2, b_2]$, then, assuming we have no outliers, the eigenvalues of X_m are approximately equal to

$$\lambda_i\Big(\mathbf{f}\Big(a_1 + j_1\frac{b_1-a_1}{m}, a_2 + j_2\frac{b_2-a_2}{m}\Big)\Big), \quad j_1, j_2 = 1, \ldots, m, \quad i = 1, \ldots, s,$$

for m large enough; and so on for $\ell \geq 3$.

Theorem 2.1 *Let $q \geq 0$ be an integer, let $\boldsymbol{p} \in \mathbb{N}^d$ and $\mathbf{0} \leq \boldsymbol{k} \leq \boldsymbol{p} - \mathbf{1}$. Assume that $K(\mathbf{x})$ is symmetric positive definite at every point $\mathbf{x} \in (0, 1)^d$ and each component of $K(\mathbf{x})$ is a continuous bounded function on $(0, 1)^d$. Suppose the following two conditions are met:*

- *$\boldsymbol{n} = \boldsymbol{\alpha} n$, where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d)$ is a vector with positive components in $\mathbb{Q}^d$ and n varies in some infinite subset of $\mathbb{N}$ such that $\boldsymbol{n} = \boldsymbol{\alpha} n \in \mathbb{N}^d$;*
- *$N = N(n)$ is such that $N \to \infty$ and $N/n^2 \to 0$ as $n \to \infty$.*

Then, for the sequence of normalized space-time matrices $\{2Nn^{d-2} C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)\}_n$ we have the spectral distribution relation

$$\{2Nn^{d-2} C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)\}_n \sim_\lambda \mathbf{f}_{[q,\boldsymbol{p},\boldsymbol{k}]}^{[\boldsymbol{\alpha},K]},$$

where:

- *the spectral symbol $\mathbf{f}_{[q,\boldsymbol{p},\boldsymbol{k}]}^{[\boldsymbol{\alpha},K]} : [0, 1]^d \times [-\pi, \pi]^d \to \mathbb{C}^{(q+1)\prod_{i=1}^d (p_i - k_i)\times(q+1)\prod_{i=1}^d (p_i-k_i)}$ is defined as*

$$\mathbf{f}_{[q,\boldsymbol{p},\boldsymbol{k}]}^{[\boldsymbol{\alpha},K]}(\mathbf{x}, \boldsymbol{\theta}) = \mathbf{f}_{[\boldsymbol{p},\boldsymbol{k}]}^{[\boldsymbol{\alpha},K]}(\mathbf{x}, \boldsymbol{\theta}) \otimes \mathrm{T}M_{[q]}; \tag{2.10}$$

- *$\mathbf{f}_{[\boldsymbol{p},\boldsymbol{k}]}^{[\boldsymbol{\alpha},K]} : [0, 1]^d \times [-\pi, \pi]^d \to \mathbb{C}^{\prod_{i=1}^d (p_i-k_i)\times\prod_{i=1}^d (p_i-k_i)}$ is defined as*

$$\mathbf{f}_{[\boldsymbol{p},\boldsymbol{k}]}^{[\boldsymbol{\alpha},K]}(\mathbf{x}, \boldsymbol{\theta}) = \frac{1}{\prod_{i=1}^d \alpha_i} \sum_{i,j=1}^{d} \alpha_i \alpha_j K_{ij}(\mathbf{x})(H_{[\boldsymbol{p},\boldsymbol{k}]})_{ij}(\boldsymbol{\theta}); \tag{2.11}$$

– $H_{[\boldsymbol{p},\boldsymbol{k}]}$ *is a* $d \times d$ *block matrix whose* (i, j) *entry is a* $\prod_{i=1}^{d}(p_i - k_i) \times \prod_{i=1}^{d}(p_i - k_i)$ *block defined as in* [10, *eq.* (5.12)]*;*
– T *is the final time in* (1.1) *and* $M_{[q]}$ *is given in* (2.7)*.*

With the same argument used for proving Theorem 2.1, it not difficult to prove the following result.

Theorem 2.2 *Suppose the hypotheses of Theorem 2.1 are satisfied, and let*

$$Q_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K) = \frac{\Delta t}{2} I_N \otimes M_{[q]} \otimes K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K).$$

Then,

$$\{2Nn^{d-2}(I_N \otimes A_{\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K))\}_n \sim_\lambda \mathbf{f}_{[q,\boldsymbol{p},\boldsymbol{k}]}^{[\boldsymbol{\alpha},K]}, \qquad \{2Nn^{d-2}Q_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)\}_n \sim_\lambda \mathbf{f}_{[q,\boldsymbol{p},\boldsymbol{k}]}^{[\boldsymbol{\alpha},K]}.$$

3 PGMRES for the Space-time IgA-DG System

Suppose the hypotheses of Theorem 2.1 are satisfied. Then, on the basis of Theorem 2.2 and the theory of (block) generalized locally Toeplitz (GLT) sequences [7,8,30,31,52,53], we expect that the sequence of preconditioned matrices

$$(I_N \otimes A_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K))^{-1} C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K), \tag{3.1}$$

as well as the sequence of preconditioned matrices

$$(Q_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K))^{-1} C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K) = \frac{2}{\Delta t} (I_N \otimes M_{[q]} \otimes K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K))^{-1} C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K), \tag{3.2}$$

has an asymptotic spectral distribution described by the preconditioned symbol

$$\left(\mathbf{f}_{[q,\boldsymbol{p},\boldsymbol{k}]}^{[\boldsymbol{\alpha},K]}\right)^{-1} \mathbf{f}_{[q,\boldsymbol{p},\boldsymbol{k}]}^{[\boldsymbol{\alpha},K]} = I_{(q+1)\prod_{i=1}^{d}(p_i-k_i)}.$$

This means that the eigenvalues of the two sequences of matrices (3.1) and (3.2) are (weakly) clustered at 1; see [7, Sect. 2.4.2]. Therefore, in view of the convergence properties of the GMRES method [13]—see in particular [13, Theorem 2.13] and the original research paper by Bertaccini and Ng [14]—we may expect that the PGMRES with preconditioner $I_N \otimes A_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$ or $Q_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$ for solving a linear system with coefficient matrix $C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$ has an optimal convergence rate, i.e., the number of iterations for reaching a preassigned accuracy ε is independent of (or only weakly dependent on) the matrix size. We may also expect that the same is true for the PGMRES with preconditioner

$$P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K) = I_N \otimes I_{q+1} \otimes K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K) = I_{N(q+1)} \otimes K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K), \tag{3.3}$$

because (up to a negligible normalization factor $\Delta t/2$) $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$ is spectrally equivalent to $Q_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$. Indeed, the spectrum of $(P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K))^{-1}(I_N \otimes M_{[q]} \otimes K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K))$ is contained in $[c_q, C_q]$ for some positive constants $c_q, C_q > 0$ depending only on q. For instance, one can take $c_q = \lambda_{\min}(M_{[q]})$ and $C_q = \lambda_{\max}(M_{[q]})$, which are both positive as $M_{[q]}$ is symmetric positive definite (see (2.7)).

To show that our expectation is realized, we solve system (2.1) in two space dimensions ($d = 2$), up to a precision $\varepsilon = 10^{-8}$, by means of the GMRES and the PGMRES with preconditioner $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$, using $f(t, \mathbf{x}) = 1$, $\mathrm{T} = 1$, $\boldsymbol{\alpha} = (1, 1)$, $\boldsymbol{n} = \boldsymbol{\alpha}n = (n, n)$, $\boldsymbol{p} = (p, p)$,

Table 1 Number of iterations GM[p] and PGM[p] needed by, respectively, the GMRES and the PGMRES with preconditioner $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$, for solving the linear system (2.1), up to a precision $\varepsilon = 10^{-8}$, in the case where $d = 2$, $K(\mathbf{x}) = I_2$, $f(t,\mathbf{x}) = 1$, $\mathrm{T} = 1$, $q = 0$, $\boldsymbol{n} = (n,n)$, $\boldsymbol{p} = (p,p)$, $\boldsymbol{k} = (p-1,p-1)$, $N = n$

$n = N$	GM[3]	PGM[3]	GM[4]	PGM[4]	GM[5]	PGM[5]
20	66	21	85	21	170	21
40	168	40	178	40	235	40
60	295	59	314	59	360	59
80	443	77	473	77	506	77
100	609	94	652	94	699	94
120	790	111	847	111	909	111
$n = N$	GM[6]	PGM[6]	GM[7]	PGM[7]	GM[8]	PGM[8]
20	269	21	532	21	674	21
40	380	40	572	40	656	40
60	477	59	611	59	690	59
80	621	77	720	77	791	77
100	780	94	879	94	963	94
120	971	111	1025	111	1114	111

The total size of the space-time system (number of DoFs) is given by $n\bar{n} = n(n+p-2)^2$

$\boldsymbol{k} = (k,k)$, and varying $K(\mathbf{x})$, N, n, q, p, k. The resulting number of iterations are collected in Tables 1, 2, 3. We see from the tables that the GMRES solver rapidly deteriorates with increasing n, and it is not robust with respect to p, k. On the other hand, the convergence rate of the proposed PGMRES is robust with respect to all spatial parameters n, p, k, though its performance is clearly better in the case where N is fixed (Tables 2, 3) than in the case where N increases (Table 1). An explanation of this phenomenon based on Theorem 2.1 is the following. In the case where N is fixed, the ratio N/n^2 converges to 0 much more quickly than in the case where $N = n$. Consequently, when N is fixed, the spectrum of both $2Nn^{d-2}C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$ and $2Nn^{d-2}Q_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$ is better described by the symbol $\mathbf{f}_{[q,\boldsymbol{p},\boldsymbol{k}]}^{[\boldsymbol{\alpha},K]}$ than when $N = n$. Similarly, the spectrum of the preconditioned matrix $(Q_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K))^{-1}C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$ is better described by the preconditioned symbol $I_{(q+1)\prod_{i=1}^{d}(p_i-k_i)}$. In conclusion, the eigenvalues of the preconditioned matrix are supposed to be more clustered when N is fixed than when $N = n$.

In order to investigate the influence of q on the number of PGMRES iterations, we performed a further numerical experiment in Table 4. We observe that the considered PGMRES is not robust with respect to q, but the number of PGMRES iterations grows linearly with q. By comparing Tables 1 and 4, we note that the PGMRES convergence is linear with respect to both N and q. In practice, increasing q is the most convenient way to improve the temporal accuracy of the discrete solution $\mathbf{u}$; see, e.g., [12]. This is due to the superconvergence property, according to which the order of convergence in time of a q-degree DG method is $2q+1$ [19,43]. Tables 1 and 4 show that the strategy of keeping N fixed and increasing q is more convenient even in terms of performance of the proposed PGMRES.

As it is known, each PGMRES iteration requires solving a linear system with coefficient matrix given by the preconditioner $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$, and this is not required in a GMRES iteration. Thus, if we want to prove that the proposed PGMRES is fast, we have to show that we are

Table 2 Number of iterations GM$[p, k]$ and PGM$[p, k]$ needed by, respectively, the GMRES and the PGMRES with preconditioner $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$, for solving the linear system (2.1), up to a precision $\varepsilon = 10^{-8}$, in the case where $d = 2$, $K(x_1, x_2) = \begin{bmatrix} \cos(x_1) + x_2 & 0 \\ 0 & x_1 + \sin(x_2) \end{bmatrix}$, $f(t, \mathbf{x}) = 1$, T $= 1$, $q = 1$, $\boldsymbol{n} = (n, n)$, $\boldsymbol{p} = (p, p)$, $\boldsymbol{k} = (k, k)$, $N = 20$

n	GM[1, 0]	PGM[1, 0]	GM[2, 0]	PGM[2, 0]	GM[2, 1]	PGM[2, 1]	GM[3, 1]	PGM[3, 1]
20	244	42	383	42	156	42	276	42
40	502	42	778	42	314	42	560	42
60	763	42	1174	42	474	42	842	42
80	1026	42	1570	42	635	42	1146	42
100	1275	42	1966	42	796	42	1894	42
120	1608	42	2374	42	954	42	1898	42
n	GM[4, 1]	PGM[4, 1]	GM[4, 2]	PGM[4, 2]	GM[5, 2]	PGM[5, 2]	GM[5, 3]	PGM[5, 3]
20	444	42	390	42	522	42	514	42
40	759	42	565	42	721	42	643	42
60	1148	42	771	42	953	42	831	42
80	1536	42	1035	42	1337	42	1026	42
100	1909	42	1299	42	2232	42	1226	42
120	2329	42	1564	42	2390	42	1831	42

The number of DoFs is given by $40\bar{n} = 40(n(p-k)+k-1)^2$. Note that $K(x_1, x_2)$ is singular at $(x_1, x_2) = (0, 0)$

Table 3 Number of iterations GM$[p, k]$ and PGM$[p, k]$ needed by, respectively, the GMRES and the PGMRES with preconditioner $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$, for solving the linear system (2.1), up to a precision $\varepsilon = 10^{-8}$, in the case where $d = 2$, $K(x_1, x_2) = \begin{bmatrix} (2+\cos x_1)(1+x_2) & \cos(x_1+x_2)\sin(x_1+x_2) \\ \cos(x_1+x_2)\sin(x_1+x_2) & (2+\sin x_2)(1+x_1) \end{bmatrix}$, $f(t, \mathbf{x}) = 1$, T $= 1$, $q = 2$, $\boldsymbol{n} = (n, n)$, $\boldsymbol{p} = (p, p)$, $\boldsymbol{k} = (k, k)$, $N = 20$

n	GM[2, 0]	PGM[2, 0]	GM[2, 1]	PGM[2, 1]	GM[3, 0]	PGM[3, 0]	GM[3, 2]	PGM[3, 2]
20	286	40	112	40	400	40	123	40
40	579	40	228	40	809	40	224	40
60	874	40	345	40	1218	40	339	40
80	1170	40	463	40	1716	40	456	40
100	1466	40	580	40	2204	40	573	40
120	1757	40	697	40	2487	40	690	40
n	GM[4, 0]	PGM[4, 0]	GM[4, 3]	PGM[4, 3]	GM[5, 0]	PGM[5, 0]	GM[5, 4]	PGM[5, 4]
20	779	40	208	40	1460	40	396	40
40	1070	40	270	40	1982	40	419	40
60	1580	40	361	40	2376	40	466	40
80	2176	40	487	40	2733	40	531	40
100	2668	40	613	40	3559	40	657	40
120	3284	40	738	40	4565	40	791	40

The number of DoFs is given by $60\bar{n} = 60(n(p-k)+k-1)^2$

Table 4 Same setting as in Table 1 with $n = N = 20$ and $q = 0, 1, 2, 3, 4$

q	GM[3]	PGM[3]	GM[4]	PGM[4]	GM[5]	PGM[5]
0	66	21	85	21	170	21
1	122	42	154	42	280	42
2	175	64	225	64	391	64
3	222	95	289	95	464	95
4	247	115	351	115	602	116
q	GM[6]	PGM[6]	GM[7]	PGM[7]	GM[8]	PGM[8]
0	269	21	532	21	674	21
1	446	42	688	42	834	42
2	491	64	580	64	672	64
3	616	95	916	95	1103	95
4	1031	116	1927	116	5468	116

able to solve efficiently a linear system with matrix $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$. However, for the reasons explained in Sect. 4, this is not exactly the path we will follow.

Before moving on to Sect. 4, we remark that, thanks to the tensor structure (3.3), the solution of a linear system with coefficient matrix $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$ reduces to the solution of $N(q+1)$ linear systems with coefficient matrix $K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K)$. Indeed, the solution of the system $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)\mathbf{x} = \mathbf{y}$ is given by

$$\mathbf{x} = (P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K))^{-1}\mathbf{y} = (I_{N(q+1)} \otimes K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K)^{-1})\mathbf{y} = \begin{bmatrix} K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K)^{-1}\mathbf{y}_1 \\ \vdots \\ K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K)^{-1}\mathbf{y}_{N(q+1)} \end{bmatrix}, \tag{3.4}$$

where $\mathbf{y}^T = [\mathbf{y}_1^T, \ldots, \mathbf{y}_{N(q+1)}^T]$ and each $\mathbf{y}_i$ has length $\bar{n}$. It is then clear that the computation of the solution $\mathbf{x}$ is equivalent to solving the $N(q+1)$ linear systems $K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K)\mathbf{x}_i = \mathbf{y}_i$, $i = 1, \ldots, N(q+1)$. Note that the various $\mathbf{x}_i$ can be computed in parallel as the computation of $\mathbf{x}_i$ is independent of the computation of $\mathbf{x}_j$ whenever $i \neq j$.

4 Fast Solver for the Space-time IgA-DG System

From here on, we focus on the maximal smoothness case $\boldsymbol{k} = \boldsymbol{p} - \mathbf{1}$, that is, the case corresponding to the classical IgA approach. For notational simplicity, we drop the subscript/superscript $\boldsymbol{k} = \boldsymbol{p}-\mathbf{1}$, so that, for instance, the matrices $C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{p}-\mathbf{1}]}(K)$, $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{p}-\mathbf{1}]}(K)$, $K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{p}-\mathbf{1}]}(K)$ will be denoted by $C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)$, $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)$, $K_{\boldsymbol{n},[\boldsymbol{p}]}(K)$, respectively.

The solver suggested in Sect. 3 for a linear system with matrix $C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)$ is a PGMRES with preconditioner $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)$. According to (3.4), the solution of a linear system with matrix $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)$, which is required at each PGMRES iteration, is equivalent to solving $N(q+1)$ linear systems with matrix $K_{\boldsymbol{n},[\boldsymbol{p}]}(K)$. Fast solvers for $K_{\boldsymbol{n},[\boldsymbol{p}]}(K)$ that have been proposed in recent papers (see [20–22] and references therein) might be employed here. However,

using an exact solver for $K_{\boldsymbol{n},[\boldsymbol{p}]}(K)$ is not what we have in mind. Indeed, it was discovered experimentally that *the PGMRES method converges faster if the linear system with matrix* $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)$ *occurring at each PGMRES iteration is solved inexactly*. More precisely, when solving the $N(q+1)$ linear systems with matrix $K_{\boldsymbol{n},[\boldsymbol{p}]}(K)$ occurring at each PGMRES iteration, *it is enough to approximate their solutions by performing only a few standard multigrid iterations in order to achieve an excellent PGMRES run-time; and, in fact, only one standard multigrid iteration is sufficient*. In view of these experimental discoveries, we propose to solve a linear system with matrix $C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)$ in the following way.

Algorithm 1

1. *Apply to the given system the PGMRES algorithm with preconditioner* $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)$.
2. *The exact solution of the linear system with matrix* $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)$ *occurring at each PGMRES iteration would require solving* $N(q+1)$ *linear systems with matrix* $K_{\boldsymbol{n},[\boldsymbol{p}]}(K)$ *as per eq.* (3.4).
3. *Instead of solving exactly these* $N(q+1)$ *systems, apply to each of them, starting from the zero vector as initial guess,* μ *multigrid (V-cycle) iterations involving, at all levels, a single symmetric Gauss–Seidel post-smoothing step and standard bisection for the interpolation and restriction operators (following the Galerkin assembly in which the interpolation operator is the transpose of the restriction operator).*

As we shall see in the numerics of Sect. 6, the choice $\mu = 1$ yields the best performance of Algorithm 1. The proposed solver is not the PGMRES with preconditioner $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)$ because, at each iteration, the linear system associated with $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)$ is not solved exactly. However, the solver is still a PGMRES with a different preconditioner $\tilde{P}_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)$. To see this, let MG be the iteration matrix of the multigrid method used in step 3 of Algorithm 1 for solving a linear system with matrix $K_{\boldsymbol{n},[\boldsymbol{p}]}(K)$. Recall that MG depends only on $K_{\boldsymbol{n},[\boldsymbol{p}]}(K)$ and not on the specific right-hand side of the system to solve. If the system to solve is $K_{\boldsymbol{n},[\boldsymbol{p}]}(K)\mathbf{x}_i = \mathbf{y}_i$, the approximate solution $\tilde{\mathbf{x}}_i$ obtained after μ multigrid iterations starting from the zero initial guess is given by

$$\tilde{\mathbf{x}}_i = (I_{\bar{n}} - MG^{\mu})K_{\boldsymbol{n},[\boldsymbol{p}]}(K)^{-1}\mathbf{y}_i.$$

Hence, the approximation $\tilde{\mathbf{x}}$ computed by our solver for the exact solution (3.4) of the system $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)\mathbf{x} = \mathbf{y}$ is given by

$$\tilde{\mathbf{x}} = \begin{bmatrix} (I_{\bar{n}} - MG^{\mu})K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K)^{-1}\mathbf{y}_1 \\ \vdots \\ (I_{\bar{n}} - MG^{\mu})K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K)^{-1}\mathbf{y}_{N(q+1)} \end{bmatrix} = (I_{N(q+1)} \otimes (I_{\bar{n}} - MG^{\mu})K_{\boldsymbol{n},[\boldsymbol{p}]}(K)^{-1})\mathbf{y}$$
$$= \tilde{P}_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)^{-1}\mathbf{y},$$

where

$$\tilde{P}_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K) = I_{N(q+1)} \otimes K_{\boldsymbol{n},[\boldsymbol{p}]}(K)(I_{\bar{n}} - MG^{\mu})^{-1}. \tag{4.1}$$

In conclusion, *the proposed solver is the PGMRES with preconditioner* $\tilde{P}_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)$. From the expression of $\tilde{P}_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)$, we can also say that *the proposed solver is a MG-GMRES, that is, a PGMRES with preconditioner given by a standard multigrid applied only in space.* A more precise notation for this solver could be MG_{space}-GMRES, but for simplicity we just write MG-GMRES.

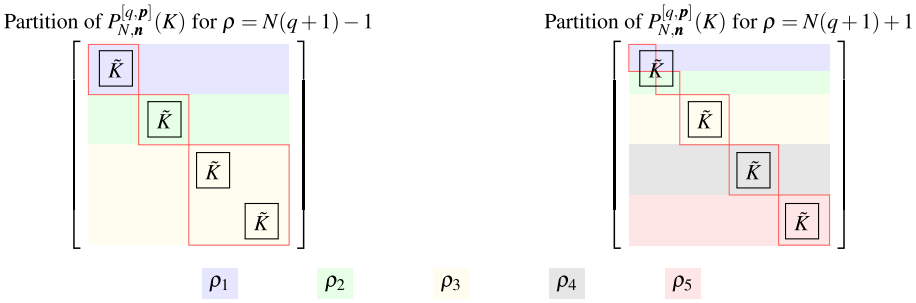

Fig. 1 Row-wise partitions of the preconditioner $P^{[q,p]}_{N,n}(K) = I_{N(q+1)} \otimes \tilde{K}$ using $\rho = N(q+1)-1$ processors (left) and $\rho = N(q+1)+1$ processors (right) with $N(q+1) = 4$. For simplicity, we write "$\tilde{K}$" instead of "$K^{[q,p]}_{N,n}(K)$"

5 Fast Parallel Solver for the Space-time IgA-DG System

In Sect. 4, we have described the sequential version of the proposed solver. The same version is used also in the case where $\rho \leq N(q+1)$ processors are available, with the only difference that step 3 of Algorithm 1 is performed in parallel. In practice, s_i linear systems are assigned to the ith processor for $i = 1, \ldots, \rho$, with $s_1 + \ldots + s_\rho = N(q+1)$ and $s_1, \ldots, s_\rho$ approximately equal to each other according to a load balancing principle. This is illustrated in Fig. 1 (left), which shows the row-wise partition of $P^{[q,p]}_{N,n}(K) = I_{N(q+1)} \otimes K^{[q,p]}_{N,n}(K)$ corresponding to the distribution of the $N(q+1)$ systems among $\rho = N(q+1)-1$ processors.

If $\rho > N(q+1)$ processors are available, we use a slight modification of the solver, which is suited for parallel computation. As before, the modification only concerns step 3 of Algorithm 1. Since we now have more processors than systems to be solved, after assigning 1 processor to each system, we still have $\rho - N(q+1)$ unused processors. Following again a load balancing principle, we distribute the unused processors among the $N(q+1)$ systems, so that now one system can be shared between two or more different processors; see Fig. 1 (right). Suppose that the system $K^{[q,p]}_{N,n}(K)\mathbf{x} = \mathbf{y}$ is shared between σ processors. The symmetric Gauss–Seidel post-smoothing iteration in step 3 of Algorithm 1 cannot be performed in parallel. Therefore, we replace it with its block-wise version. To be precise, we recall that the symmetric Gauss–Seidel iteration for a system with matrix $E = L + U - D$ is just the preconditioned Richardson iteration with preconditioner $M = LD^{-1}U$.[1] Its block-wise version in the case where we consider σ diagonal blocks $E_1, \ldots, E_\sigma$ of E is simply the preconditioned Richardson iteration with preconditioner $M_1 \oplus \cdots \oplus M_\sigma$, where M_i is the symmetric Gauss–Seidel preconditioner for E_i and $M_1 \oplus \cdots \oplus M_\sigma$ is the block diagonal matrix whose diagonal blocks are $M_1, \ldots, M_\sigma$. This block-wise version is suited for parallel computation in the case where σ processors are available.

[1] The matrices L, U, D are, respectively, the lower triangular part of E (including the diagonal), the upper triangular part of E (including the diagonal), and the diagonal part of E.

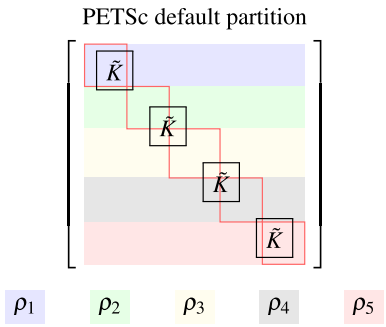

Fig. 2 The PETSc default row-wise partition does not account for the structure of the space-time problem; compare with Fig. 1

6 Numerical Experiments: Iteration Count, Timing and Scaling

In this section, we illustrate through numerical experiments the performance of the proposed solver and we compare it to the performance of other benchmark parallel solvers, such as the PGMRES with block-wise ILU(0) preconditioner.

6.1 Implementation Details

For the numerics of this section, as well as throughout this paper, we used the C++ framework PETSc [5,6] and the domain specific language Utopia [64] for the parallel linear algebra and solvers, and the Cray-MPICH compiler. For the assembly of high order finite elements, we used the PetIGA package [18]. A parallel tensor-product routine was implemented to assemble space-time matrices. Numerical experiments have been performed on the Cray XC40 nodes of the Piz Daint supercomputer of the Swiss national supercomputing centre (CSCS).[2] The used partition features 1813 computation nodes, each of which holds two 18-core Intel Xeon E5-2695v4 (2.10GHz) processors. We stress that the PETSc default row-wise partition follows a load balancing principle and, except in the trivial case $\rho = N$, does not correspond to the row-wise partition described in Sect. 5; see Fig. 2. Therefore, the partition must be adjusted by the user. Alternatively, one can use a PETSc built-in class for sparse block matrices and specify the block size $(q+1)\bar{n}$.

6.2 Experimental Setting

In the numerics of this section, we solve the linear system (2.1) arising from the choices $d = 2$, $f(t, \mathbf{x}) = 1$, $\mathrm{T} = 1$, $\boldsymbol{n} = (n, n)$, $\boldsymbol{p} = (p, p)$, $\boldsymbol{k} = (p-1, p-1)$. The basis functions $\ell_{1,[q]}, \ldots, \ell_{q+1,[q]}$ are chosen as the Lagrange polynomials associated with the right Gauss–Radau nodes in $[-1, 1]$. The values of $K(\mathbf{x})$, N, n, q, p are specified in each example. For each solver considered herein, we use the tolerance $\varepsilon = 10^{-8}$ and the PETSc default stopping criterion based on the preconditioned relative residual. Moreover, the PGMRES method is always applied with restart after 30 iterations as per PETSc default. Whenever we report the run-time of a solver, the time spent in I/O operations and matrix assembly is ignored. Run-times are always expressed in seconds. In all the tables below, the number of iterations needed by a given solver to converge within the tolerance $\varepsilon = 10^{-8}$ is reported in square

[2] https://www.cscs.ch/computers/piz-daint/

brackets next to the corresponding run-time. Throughout this section, we use the following abbreviations for the solvers.

– ILU(0)-GMRES PGMRES with preconditioner given by an ILU(0) factorization (ILU factorization with no fill-in) of the system matrix.
– $\text{MG}^{L}_{\mu,\nu}$-GMRES The proposed solver, as described in Sect. 4, with μ multigrid (V-cycle) iterations applied to $K_{\boldsymbol{n},[\boldsymbol{p}]}(K)$. Each multigrid iteration involves ν symmetric Gauss–Seidel post-smoothing steps at the finest level and 1 symmetric Gauss–Seidel post-smoothing step at the coarse levels. The choice $\nu = 1$ corresponds to our solver proposal. Different values of ν are considered for comparison purposes. The superscript L denotes the number of multigrid levels.
– $\text{TMG}^{L}_{\mu,\nu}$-GMRES The same as $\text{MG}^{L}_{\mu,\nu}$-GMRES, with the only difference that the multigrid iterations are performed with the telescopic option, thus giving rise to the telescopic multigrid (TMG) [23,45]. This technique consists in reducing the number of processors used on the coarse levels and can be beneficial for the parallel multigrid performance. In the numerics of this section, we only reduced the number of processors used on the coarsest level to one fourth of the number of processors used at all other levels.

6.3 Iteration Count and Timing

Tables 5, 6, 7 illustrate the performance of the proposed solver in terms of number of iterations and run-time. It is clear from the tables that the best performance of the solver is obtained when applying to $K_{\boldsymbol{n},[\boldsymbol{p}]}(K)$ a single multigrid iteration ($\mu = 1$) with only one smoothing step at the finest level ($\nu = 1$). Moreover, the solver is competitive with respect to the ILU(0)-GMRES. The worst performance of the solver with respect to the ILU(0)-GMRES is attained in Table 6, where the diffusion matrix $K(x_1, x_2)$ is singular at $(x_1, x_2) = (0, 0)$.

Table 5 PGMRES iterations and run-time (using 64 cores) to solve the linear system (2.1) up to a precision of 10^{-8}, according to the experimental setting described in Sect. 6.2

p	1	2	3	4	
ILU(0)-GMRES	3.7 [579]	4.3 [367]	5.2 [269]	6.7 [226]	
$\text{MG}^{5}_{3,2}$-GMRES	1.4 [33]	2.9 [33]	4.7 [33]	7.2 [33]	
$\text{MG}^{5}_{1,2}$-GMRES	0.8 [33]	1.6 [33]	2.5 [33]	4.0 [35]	
$\text{MG}^{5}_{3,1}$-GMRES	1.1 [33]	2.2 [33]	3.3 [33]	5.0 [34]	
$\text{MG}^{5}_{1,1}$-GMRES	0.6 [33]	1.2 [33]	1.8 [34]	3.1 [39]	
p	5	6	7	8	9
ILU(0)-GMRES	8.2 [193]	10.1 [174]	11.9 [156]	22.5 [234]	44.9 [383]
$\text{MG}^{5}_{3,2}$-GMRES	10.5 [35]	14.7 [36]	21.1 [41]	34.6 [53]	57.6 [73]
$\text{MG}^{5}_{1,2}$-GMRES	6.6 [42]	11.0 [52]	16.0 [60]	26.2 [77]	47.0 [90]
$\text{MG}^{5}_{3,1}$-GMRES	7.1 [36]	11.4 [43]	17.0 [51]	28.5 [67]	42.1 [83]
$\text{MG}^{5}_{1,1}$-GMRES	5.3 [50]	9.1 [63]	13.5 [75]	19.8 [87]	30.7 [112]

We used $K(\mathbf{x}) = I_2$, $q = 0$, $N = 32$, $n = 259 - p$. The total size of the space-time system (number of DoFs) is given by $32 \cdot 257^2$

Table 6 PGMRES iterations and run-time (using 64 cores) to solve the linear system (2.1) up to a precision of 10^{-8}, according to the experimental setting described in Sect. 6.2

p	1	2	3	4	
ILU(0)-GMRES	1.3 [449]	1.7 [283]	2.2 [219]	2.9 [183]	
$MG^5_{2,3}$-GMRES	0.6 [55]	1.3 [55]	2.4 [55]	4.1 [58]	
$MG^5_{1,3}$-GMRES	0.5 [57]	1.0 [56]	1.8 [56]	3.5 [68]	
$MG^5_{2,1}$-GMRES	0.5 [57]	1.0 [57]	1.6 [58]	3.1 [77]	
$MG^5_{1,1}$-GMRES	0.5 [67]	0.8 [65]	1.3 [68]	2.8 [90]	
p	5	6	7	8	9
ILU(0)-GMRES	3.6 [158]	4.4 [141]	6.0 [148]	9.5 [186]	24.8 [397]
$MG^5_{2,3}$-GMRES	7.6 [64]	12.7 [90]	18.5 [101]	32.2 [139]	48.9 [173]
$MG^5_{1,3}$-GMRES	6.2 [85]	10.4 [103]	15.0 [116]	26.5 [161]	38.0 [189]
$MG^5_{2,1}$-GMRES	5.2 [91]	8.6 [112]	12.6 [128]	22.0 [179]	30.7 [205]
$MG^5_{1,1}$-GMRES	4.6 [110]	7.2 [125]	11.0 [150]	19.4 [210]	30.2 [269]

We used $K(x_1, x_2) = \begin{bmatrix} \cos(x_1) + x_2 & 0 \\ 0 & x_1 + \sin(x_2) \end{bmatrix}$, $q = 1$, $N = 20$, $n = 131 - p$. The total size of the space-time system (number of DoFs) is given by $40 \cdot 129^2$. Note that $K(x_1, x_2)$ is singular at $(x_1, x_2) = (0, 0)$

Table 7 PGMRES iterations and run-time (using 64 cores) to solve the linear system (2.1) up to a precision of 10^{-8}, according to the experimental setting described in Sect. 6.2

p	1	2	3	4	
ILU(0)-GMRES	1.9 [450]	2.2 [284]	2.6 [205]	3.4 [170]	
$MG^5_{2,2}$-GMRES	0.2 [11]	0.5 [11]	0.8 [11]	1.5 [13]	
$MG^5_{1,2}$-GMRES	0.2 [12]	0.4 [11]	0.6 [12]	1.2 [15]	
$MG^5_{2,1}$-GMRES	0.2 [11]	0.4 [11]	0.6 [12]	1.1 [15]	
$MG^5_{1,1}$-GMRES	0.2 [12]	0.3 [11]	0.5 [14]	1.0 [19]	
p	5	6	7	8	9
ILU(0)-GMRES	4.4 [154]	5.2 [135]	6.4 [125]	12.6 [195]	22.8 [289]
$MG^5_{2,2}$-GMRES	2.6 [17]	4.1 [20]	5.9 [23]	8.8 [27]	11.9 [30]
$MG^5_{1,2}$-GMRES	2.1 [20]	3.3 [23]	4.6 [26]	7.2 [31]	10.1 [36]
$MG^5_{2,1}$-GMRES	2.0 [20]	3.1 [23]	4.6 [27]	6.2 [31]	8.4 [35]
$MG^5_{1,1}$-GMRES	1.7 [23]	2.5 [26]	3.6 [30]	5.5 [36]	7.4 [40]

We used $K(x_1, x_2) = \begin{bmatrix} (2 + \cos x_1)(1 + x_2) & \cos(x_1 + x_2)\sin(x_1 + x_2) \\ \cos(x_1 + x_2)\sin(x_1 + x_2) & (2 + \sin x_2)(1 + x_1) \end{bmatrix}$, $q = 0$, $N = 20$, $n = 259 - p$. The total size of the space-time system (number of DoFs) is given by $20 \cdot 257^2$

Table 8 Strong scaling: PGMRES iterations and run-time to solve the linear system (2.1) up to a precision of 10^{-8}, according to the experimental setting described in Sect. 6.2.

Cores	1	2	4	8	
ILU(0)-GMRES	1385.0 [414]	682.1 [415]	336.7 [415]	181.9 [415]	
$MG^{7}_{1,1}$-GMRES	335.1 [64]	179.7 [64]	92.5 [64]	51.9 [64]	
$TMG^{7}_{1,1}$-GMRES	335.1 [64]	179.7 [64]	92.5 [64]	51.9 [64]	
Cores	16	32	64	128	
ILU(0)-GMRES	103.3 [415]	49.7 [416]	21.2 [417]	12.8 [500]	
$MG^{7}_{1,1}$-GMRES	31.8 [64]	16.6 [64]	8.3 [64]	4.4 [64]	
$TMG^{7}_{1,1}$-GMRES	31.3 [64]	16.5 [64]	8.0 [64]	4.2 [64]	
Cores	256	512	1024	2048	4096
ILU(0)-GMRES	6.8 [519]	4.0 [550]	2.5 [619]	1.7 [753]	1.7 [1013]
$MG^{7}_{1,1}$-GMRES	2.5 [65]	1.8 [65]	1.9 [65]	5.1 [65]	14.7 [66]
$TMG^{7}_{1,1}$-GMRES	2.2 [64]	1.3 [63]	0.8 [64]	0.5 [64]	0.4 [64]

We used $K(\mathbf{x}) = I_2$, $q = 0$, $p = 3$, $N = 64$, $n = 384$. The total size of the space-time system (number of DoFs) is given by $64 \cdot 385^2$

Table 9 Space-time weak scaling: PGMRES iterations and run-time to solve the linear system (2.1) up to a precision of 10^{-8}, according to the experimental setting described in Sect. 6.2

[Cores, n, N, L]	[1, 65, 8, 4]	[8, 129, 16, 5]	[64, 257, 32, 6]	[512, 513, 64, 7]
ILU(0)-GMRES	0.25 [50]	0.86 [121]	2.80 [367]	7.6 [989]
$TMG^{L}_{1,1}$-GMRES	0.11 [10]	0.27 [17]	0.67 [33]	1.4 [64]

We used $K(\mathbf{x}) = I_2, q = 0$, $p = 2$ and $(N, n) = (8, 65), (16, 129), (32, 256), (64, 512)$. The ratio DoFs/Cores is constant in the table

6.4 Scaling

In the scaling experiments, besides the multigrid already considered above, we also employ a TMG for performance reasons. To avoid memory bounds, we use at most 16 cores per node. From Table 8 and Fig. 3 we see that the proposed solver, especially when using the TMG option, shows a nearly optimal strong scaling with respect to the number of cores.[3] Table 9 and Fig. 4 illustrate the weak scaling properties of the proposed solver, which possesses a superior parallel efficiency with respect to the standard ILU(0) approach in terms of iteration count and run-time. For both solvers, however, the weak scaling is not ideal (constant run-time). This is due to the fact that N grows from 8 to 64 and both solvers are not robust with respect to N.

[3] We observe a slight reduction in the ideal scaling as the number of cores grows from 2 to 16. This is due to the fact that runs are performed on a single node with its own limited memory. For more than 16 cores, the memory bound is no longer present since computations are performed on multiple nodes with increasing memory. When the number of cores exceeds two thousand, communication takes over and scaling is no longer observable.

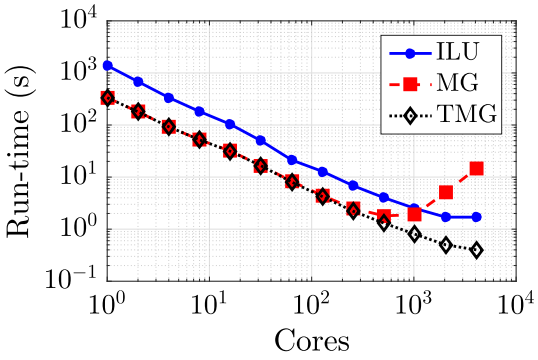

Fig. 3 Graphical representation of the run-times reported in Table 8

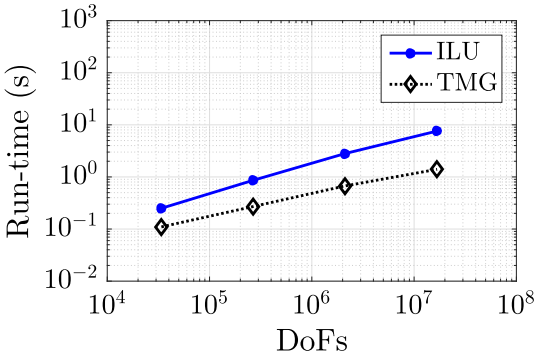

Fig. 4 Graphical representation of the run-times reported in Table 9

7 Non-rectangular Domain and Non-trivial Geometry

So far, the performance of the proposed solver has been illustrated for the diffusion problem (1.1) over the hypersquare $(0, 1)^d$. However, no special difficulty arises if $(0, 1)^d$ is replaced by a non-rectangular domain Ω described (exactly) by a geometry map $\mathbf{G} : [0, 1]^d \to \overline{\Omega}$ as per IgA paradigm. Indeed, as long as a tensor-product structure between space and time is maintained, the geometry map $\mathbf{G}$ acts as a reparameterization of Ω through $(0, 1)^d$, and the resulting discretization matrix is still given by (2.2)–(2.9) with the only difference that:

- a factor $|\det(J_{\mathbf{G}}(\mathbf{x}))|$ should be included in the integrand of (2.5), where $J_{\mathbf{G}}(\mathbf{x})$ is the Jacobian matrix of $\mathbf{G}(\mathbf{x})$;
- the matrix $K(\mathbf{x})$ in (2.6) should be replaced by $J_{\mathbf{G}}(\mathbf{x})^{-1} K(\mathbf{G}(\mathbf{x})) J_{\mathbf{G}}(\mathbf{x})^{-T} |\det(J_{\mathbf{G}}(\mathbf{x}))|$.

In short, a change of domain from $(0, 1)^d$ to Ω essentially amounts to a mere change of diffusion matrix from K to $J_{\mathbf{G}}^{-1} K(\mathbf{G}) J_{\mathbf{G}}^{-T} |\det(J_{\mathbf{G}})|$, which does not affect the performance of the proposed solver.

In Table 10, we validate the previous claim by testing the solver on the linear system arising from the space-time IgA-DG discretization of (1.1) in the case where $(0, 1)^d$ is replaced by a non-rectangular domain Ω described by a non-trivial geometry map $\mathbf{G} : [0, 1]^d \to \overline{\Omega}$. The experimental setting is the same as in Sect. 6.2, with the only difference that $(0, 1)^2$ is now replaced by a quarter of an annulus

$$\Omega = \{\mathbf{x} \in \mathbb{R}^2 : r^2 < x_1^2 + x_2^2 < R^2,\ x_1 > 0,\ x_2 > 0\}, \qquad r = 1, \qquad R = 2, \tag{7.1}$$

Table 10 PGMRES iterations and run-time (using 64 cores) to solve, up to a precision of 10^{-8}, the linear system arising from the space-time IgA-DG discretization of (1.1) in the case where $(0, 1)^d$ is replaced by the domain (7.1) described by the geometry map (7.2). The experimental setting is the same as in Sect. 6.2

p	1	2	3	4	
ILU(0)-GMRES	1.6 [412]	2.2 [354]	4.0 [296]	4.3 [268]	
$MG^5_{1,1}$-GMRES	0.7 [99]	1.3 [91]	2.2 [103]	4.0 [140]	
p	5	6	7	8	9
ILU(0)-GMRES	6.0 [266]	8.0 [257]	16.6 [415]	31.3 [622]	48.2 [775]
$MG^5_{1,1}$-GMRES	7.2 [178]	11.8 [219]	16.5 [241]	29.6 [348]	39.9 [386]

We used $K(\mathbf{x}) = I_2$, $q = 1$, $N = 20$, $n = 131 - p$. The total size of the space-time system (number of DoFs) is given by $40 \cdot 129^2$

described by the geometry map $\mathbf{G} : [0, 1]^2 \to \overline{\Omega}$,

$$\mathbf{G}(\hat{\mathbf{x}}) = \begin{cases} x_1 = [r + \hat{x}_1(R - r)] \cos\left(\frac{\pi}{2}\hat{x}_2\right), \\ x_2 = [r + \hat{x}_1(R - r)] \sin\left(\frac{\pi}{2}\hat{x}_2\right), \end{cases} \qquad \hat{\mathbf{x}} \in [0, 1]^2. \tag{7.2}$$

We remark that the geometry map $\mathbf{G}$ is a common benchmark example in IgA; see, e.g., [20,21].

8 Conclusions

We have proposed a MG-GMRES solver for the space-time IgA-DG discretization of the diffusion problem (1.1). Through numerical experiments, we have illustrated the competitiveness of our proposal in terms of iteration count, run-time and parallel scaling. We have also shown its applicability to more general problems than (1.1) involving a non-rectangular domain Ω and a non-trivial geometry map $\mathbf{G}$. To conclude, we remark that the proposed solver is highly flexible as it does not depend on the domain or the space-time discretization. It could therefore be applied to other space-time discretizations, as long as a tensor-product structure is maintained between space and time.

Acknowledgements Paola Ferrari is partially financed by the GNCS 2019 Project "Metodi Numerici per Problemi Mal Posti" . Paola Ferrari, Carlo Garoni and Stefano Serra-Capizzano are grateful to the Italian INdAM-GNCS for the scientific support. Carlo Garoni acknowledges the MIUR Excellence Department Project awarded to the Department of Mathematics of the University of Rome Tor Vergata (CUP E83C18000100006) and the support obtained by the Beyond Borders Programme of the University of Rome Tor Vergata through the Project ASTRID (CUP E84I19002250005). Rolf Krause acknowledges the funding obtained from the European High-Performance Computing Joint Undertaking (JU) under Grant Agreement N. 955701 (Project TIME-X); the JU receives support from the European Union's Horizon 2020 Research and Innovation Programme and from Belgium, France, Germany and Switzerland. Finally, the authors acknowledge the Deutsche Forschungsgemeinschaft (DFG) as part of the "ExaSolvers" Project in the Priority Programme 1648 "Software for Exascale Computing" (SPPEXA) and the Swiss National Science Foundation (SNSF) under the lead agency grant agreement SNSF-162199.

Funding Open access funding provided by University of Rome Tor Vergata within the CRUI-CARE Agreement.

Data Availability If requested by the handling editor or the reviewers, the codes used for producing the numerical results of this paper will be made publicly available.

References

1. Abedi, R., Petracovici, B., Haber, R.B.: A space-time discontinuous Galerkin method for linearized elastodynamics with element-wise momentum balance. Comput. Methods Appl. Mech. Engrg. **195**, 3247–3273 (2006)
2. Arbenz, P., Hupp, D.: Obrist, D.: A parallel solver for the time-periodic Navier–Stokes equations. In "PPAM 2013: Parallel Processing and Applied Mathematics". Springer (2014), pp. 291–300
3. Auricchio, F., Beirão da Veiga, L., Hughes, T.J.R., Reali, A., Sangalli, G.: Isogeometric collocation methods. Math. Models Methods Appl. Sci. **20**, 2075–2107 (2010)
4. Aziz, A.K., Monk, P.: Continuous finite elements in space and time for the heat equation. Math. Comput. **52**, 255–274 (1989)
5. Balay, S., Abhyankar, S., Adams, M.F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Dener, A., Eijkhout, V., Gropp, W.D., Karpeyev, D., Kaushik, D., Knepley, M.G., May, D.A., Curfman McInnes, L., Mills, R.T., Munson, T., Rupp, K., Sanan, P., Smith, B.F., Zampini, S., Zhang, H., Zhang, H.: PETSc web page. https://www.mcs.anl.gov/petsc (2019)
6. Balay, S., Abhyankar, S., Adams, M.F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Dener, A., Eijkhout, V., Gropp, W.D., Karpeyev, D., Kaushik, D., Knepley, M.G., May, D.A., Curfman McInnes, L., Mills, R.T., Munson, T., Rupp, K., Sanan, P., Smith, B.F., Zampini, S., Zhang, H., Zhang, H.: PETSc users manual. Technical Report ANL-95/11 - Revision 3.11, Argonne National Laboratory (2019)
7. Barbarino, G., Garoni, C., Serra-Capizzano, S.: Block generalized locally Toeplitz sequences: theory and applications in the unidimensional case. Electron. Trans. Numer. Anal. **53**, 28–112 (2020)
8. Barbarino, G., Garoni, C., Serra-Capizzano, S.: Block generalized locally Toeplitz sequences: theory and applications in the multidimensional case. Electron. Trans. Numer. Anal. **53**, 113–216 (2020)
9. Beirão da Veiga, L., Buffa, A., Sangalli, G., Vázquez, R.: Mathematical analysis of variational isogeometric methods. Acta Numerica **23**, 157–287 (2014)
10. Benedusi, P., Garoni, C., Krause, R., Li, X., Serra-Capizzano, S.: Space-time FE-DG discretization of the anisotropic diffusion equation in any dimension: the spectral symbol. SIAM J. Matrix Anal. Appl. **39**, 1383–1420 (2018)
11. Benedusi, P., Hupp, D., Arbenz, P., Krause, R.: A parallel multigrid solver for time-periodic incompressible Navier–Stokes equations in 3D. In "Numerical Mathematics and Advanced Applications ENUMATH 2015", Springer (2016), pp. 265–273
12. Benedusi, P., Minion, M., Krause, R.: An experimental comparison of a space-time multigrid method with PFASST for a reaction-diffusion problem. Comput. Math. Appl. (in press)
13. Bertaccini, D., Durastante, F.: Iterative Methods and Preconditioning for Large and Sparse Linear Systems with Applications. Taylor & Francis, Boca Raton (2018)
14. Bertaccini, D., Ng, M.K.: Band-Toeplitz preconditioned GMRES iterations for time-dependent PDEs. BIT Numer. Math. **43**, 901–914 (2003)
15. Betsch, P., Steinmann, P.: Conservation properties of a time FE method–part II: time-stepping schemes for non-linear elastodynamics. Inter. J. Numer. Methods Engrg. **50**, 1931–1955 (2001)
16. Česenek, J., Feistauer, M.: Theory of the space-time discontinuous Galerkin method for nonstationary parabolic problems with nonlinear convection and diffusion. SIAM J. Numer. Anal. **50**, 1181–1206 (2012)
17. Cottrell, J.A., Hughes, T.J.R., Bazilevs, Y.: Isogeometric Analysis: Toward Integration of CAD and FEA. Wiley, Chicester (2009)
18. Dalcin, L., Collier, N., Vignal, P., Côrtes, A.M.A., Calo, V.M.: PetIGA: a framework for high performance isogeometric analysis. Comput. Methods Appl. Mech. Engrg. **308**, 151–181 (2016)
19. Delfour, M., Hager, W., Trochu, F.: Discontinuous Galerkin methods for ordinary differential equations. Math. Comput. **36**, 455–473 (1981)
20. Donatelli, M., Garoni, C., Manni, C., Serra-Capizzano, S., Speleers, H.: Robust and optimal multi-iterative techniques for IgA Galerkin linear systems. Comput. Methods Appl. Mech. Engrg. **284**, 230–264 (2015)

21. Donatelli, M., Garoni, C., Manni, C., Serra-Capizzano, S., Speleers, H.: Robust and optimal multi-iterative techniques for IgA collocation linear systems. Comput. Methods Appl. Mech. Engrg. **284**, 1120–1146 (2015)
22. 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. **55**, 31–62 (2017)
23. Douglas C.C. A review of numerous parallel multigrid methods. In "Applications on Advanced Architecture Computers", SIAM (1996), pp. 187–202
24. Eriksson, K., Johnson, C., Logg, A.: Adaptive computational methods for parabolic problems: Part 1. Fundamentals. Encyclop. Comput. Mech. (2004)
25. Falgout, R.D., Friedhoff, S., Kolev, T., MacLachlan, S.P., Schroder, J.B., Vandewalle, S.: Multigrid methods with space-time concurrency. Comput. Visual. Sci. **18**, 123–143 (2007)
26. French, D.A.: A space-time finite element method for the wave equation. Comput. Methods Appl. Mech. Engrg. **107**, 145–157 (1993)
27. Gander, M.J.: 50 years of time parallel time integration. Article 3 in "Multiple Shooting and Time Domain Decomposition Methods", Springer (2015)
28. Gander, M.J., Halpern, L.: Techniques for locally adaptive time stepping developed over the last two decades. Domain Decomp. Methods Sci. Engrg. XX, 377–385 (2013)
29. Gander, M.J., Neumüller, M.: Analysis of a new space-time parallel multigrid algorithm for parabolic problems. SIAM J. Sci. Comput. **38**, A2173–A2208 (2016)
30. Garoni, C., Serra-Capizzano, S.: Generalized Locally Toeplitz Sequences: Theory and Applications, vol. I. Springer, Cham (2017)
31. Garoni, C., Serra-Capizzano, S.: Generalized Locally Toeplitz Sequences: Theory and Applications, vol. II. Springer, Cham (2018)
32. Griebel, M., Oeltz, D.: A sparse grid space-time discretization scheme for parabolic problems. Computing **81**, 1–34 (2007)
33. Hesthaven, J.S., Warburton, T.: Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications. Springer, New York (2008)
34. Hofer, C., Langer, U., Neumüller, M.: Parallel and robust preconditioning for space-time isogeometric analysis of parabolic evolution problems. SIAM J. Sci. Comput. **41**, A1793–A1821 (2019)
35. Horton, G., Vandewalle, S.: A space-time multigrid method for parabolic partial differential equations. SIAM J. Sci. Comput. **16**, 848–864 (1995)
36. Hughes, T.J.R., Cottrell, J.A., Bazilevs, Y.: Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Comput. Methods Appl. Mech. Engrg. **194**, 4135–4195 (2005)
37. Hughes, T.J.R., Hulbert, G.M.: Space-time finite element methods for elastodynamics: formulations and error estimates. Comput. Methods Appl. Mech. Engrg. **66**, 339–363 (1988)
38. Klaij, C.M., van der Vegt, J.J.W., van der Ven, H.: Space-time discontinuous Galerkin method for the compressible Navier–Stokes equations. J. Comput. Phys. **217**, 589–611 (2006)
39. Krause, D., Krause, R.: Enabling local time stepping in the parallel implicit solution of reaction-diffusion equations via space-time finite elements on shallow tree meshes. Appl. Math. Comput. **277**, 164–179 (2016)
40. Ladyženskaja, O.A., Solonnikov, V.A., Uralceva, N.N.: Linear and Quasi-Linear Equations of Parabolic Type. Amer. Math. Soc. (1968)
41. Langer, U., Moore, S.E., Neumüller, M.: Space-time isogeometric analysis of parabolic evolution problems. Comput. Methods Appl. Mech. Engrg. **306**, 342–363 (2016)
42. Langer, U., Zank, M.: Efficient direct space-time finite element solvers for parabolic initial-boundary value problems in anisotropic Sobolev spaces. arXiv:2008.01996 (2020)
43. Lasaint, P., Raviart, P.A.: On a finite element method for solving the neutron transport equation. In "Mathematical Aspects of Finite Elements in Partial Differential Equations", Academic Press (1974), pp. 89–123
44. Loli, G., Montardini, M., Sangalli, G., Tani, M.: An efficient solver for space-time isogeometric Galerkin methods for parabolic problems. Comput. Math. Appl. **80**, 2586–2603 (2020)
45. May, D.A., Sanan, P., Rupp, K., Knepley, M.G., Smith, B.F.: Extreme-scale multigrid components within PETSc. Article 5 in "Proceedings of the Platform for Advanced Scientific Computing Conference", ACM (2016)
46. McDonald, E., Wathen, A.: A simple proposal for parallel computation over time of an evolutionary process with implicit time stepping. In "Numerical Mathematics and Advanced Applications ENUMATH 2015", Springer (2016), pp. 285–293
47. Meidner, D., Vexler, B.: Adaptive space-time finite element methods for parabolic optimization problems. SIAM J. Control. Optim. **46**, 116–142 (2007)

48. Miller, S.T., Haber, R.B.: A spacetime discontinuous Galerkin method for hyperbolic heat conduction. Comput. Methods Appl. Mech. Engrg. **198**, 194–209 (2008)
49. Neumüller, M., Steinbach, O.: Refinement of flexible space-time finite element meshes and discontinuous Galerkin methods. Comput. Visual. Sci. **14**, 189–205 (2011)
50. Quarteroni, A.: Numerical Models for Differential Problems. Springer, Milan (2009)
51. Schötzau, D., Schwab, C.: An hp a priori error analysis of the DG time-stepping method for initial value problems. Calcolo **37**, 207–232 (2000)
52. Serra-Capizzano, S.: Generalized locally Toeplitz sequences: spectral analysis and applications to discretized partial differential equations. Linear Algebra Appl. **366**, 371–402 (2003)
53. Serra-Capizzano, S.: The GLT class as a generalized Fourier analysis and applications. Linear Algebra Appl. **419**, 180–233 (2006)
54. Shakib, F., Hughes, T.J.R., Zdeněk, J.: A new finite element formulation for computational fluid dynamics: X. The compressible Euler and Navier–Stokes equations. Comput. Methods Appl. Mech. Engrg. **89**, 141–219 (1991)
55. Steinbach, O., Yang, H.: Comparison of algebraic multigrid methods for an adaptive space-time finite element discretization of the heat equation in 3D and 4D. Numer. Linear Algebra Appl. **25**, e2143 (2018)
56. Steinbach, O., Yang, H.: Space-time finite element methods for parabolic evolution equations: discretization, a posteriori error estimation, adaptivity and solution. In "Space-Time Methods: Applications to Partial Differential Equations", Radon Series on Computational and Applied Mathematics 25 (2019), pp. 207–248
57. Sudirham, J.J., van der Vegt, J.J.W., van Damme, R.M.J.: Space-time discontinuous Galerkin method for advection-diffusion problems on time-dependent domains. Appl. Numer. Math. **56**, 1491–1518 (2006)
58. Tezduyar, T.E., Behr, M., Liou, J.: A new strategy for finite element computations involving moving boundaries and interfaces—The deforming-spatial-domain/space-time procedure: I. The concept and the preliminary numerical tests. Comput. Methods Appl. Mech. Engrg. **94**, 339–351 (1992)
59. Tezduyar, T.E., Behr, M., Mittal, S., Liou, J.: A new strategy for finite element computations involving moving boundaries and interfaces—The deforming-spatial-domain/space-time procedure: II. Computation of free-surface flows, two-liquid flows, and flows with drifting cylinders. Comput. Methods Appl. Mech. Engrg. **94**, 353–371 (1992)
60. Tezduyar, T.E., Sathe, S., Keedy, R., Stein, K.: Space-time finite element techniques for computation of fluid-structure interactions. Comput. Methods Appl. Mech. Engrg. **195**, 2002–2027 (2006)
61. Thite, S.: Adaptive spacetime meshing for discontinuous Galerkin methods. Comput. Geom. **42**, 20–44 (2009)
62. Thomée, V.: Galerkin Finite Element Methods for Parabolic Problems. Springer, New York (2006)
63. van der Vegt, J.J.W., van der Ven, H.: Space-time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows: I. General formulation. J. Comput. Phys. **182**, 546–585 (2002)
64. Zulian, P., Kopaničáková, A., Nestola, M.C.G., Fink, A., Fadel, N., Magri, V., Schneider, T., Botter, E., Mankau, J.: Utopia: a C++ embedded domain specific language for scientific computing. Git repository. https://bitbucket.org/zulianp/utopia (2016)