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

Pietro Benedusi
University of Italian Switzerland (USI), Institute of Computational Science,
Via Giuseppe Buffi 13, 6900 Lugano, Switzerland. Email: pietro.benedusi@usi.ch.

Paola Ferrari
University of Insubria, Department of Science and High Technology,
Via Valleggio 11, 22100 Como, Italy. Email: pferrari@uninsubria.it.

Carlo Garoni
University of Rome "Tor Vergata", Department of Mathematics,
Via della Ricerca Scientifica 1, 00133 Rome, Italy. Email: garoni@mat.uniroma2.it.

University of Insubria, Department of Science and High Technology,
Via Valleggio 11, 22100 Como, Italy. Email: carlo.garoni@uninsubria.it.

Rolf Krause
University of Italian Switzerland (USI), Institute of Computational Science,
Via Giuseppe Buffi 13, 6900 Lugano, Switzerland. Email: rolf.krause@usi.ch.

Stefano Serra-Capizzano
University of Insubria, Department of Humanities and Innovation,
Via Valleggio 11, 22100 Como, Italy. Email: stefano.serrac@uninsubria.it.

Uppsala University, Department of Information Technology, Division of Scientific Computing,
ITC, Lägerhyddsv. 2, Hus 2, P.O. Box 337, SE-751 05 Uppsala, Sweden. Email: stefano.serra@it.uu.se.

September 23, 2019

Abstract

We consider the space-time discretization of the (linear) anisotropic 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 new solution method combining a suitable preconditioned GMRES (PGMRES) algorithm with a few iterations of an appropriate multigrid method. The performance of our new solution method is illustrated through numerical experiments, which show its competitiveness in terms of robustness, run-time and parallel scaling.

Keywords: isogeometric analysis, discontinuous Galerkin, preconditioned GMRES, multigrid, parallel solver, spectral distribution, anisotropic diffusion

2010 MSC: 65M60, 65F08, 65M55, 65Y05, 47B06, 35Q79

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 [39, 53, 54, 55, 59] and allow for space-time adaptivity [1, 21, 26, 40, 45, 46, 57]. 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 [22]. Space-time methods have been used in combination with various numerical techniques, including finite differences [2, 10, 36], finite elements [4, 23, 37, 41], isogeometric analysis [35, 42], and discontinuous Galerkin methods [1, 13, 33, 37, 39, 43, 53, 59]. Moreover, they have been considered for a variety of applications, such as mechanics [12], fluid dynamics [10, 39, 52], fluid-structure interaction [56], 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 discountionus Galerkin formulation with an "upwind" flow is a common choice in this context; see, for example, [39, 49, 53, 58].

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 [28] and the parallel preconditioners for space-time isogeometric analysis proposed by Hofer et al. [35].

In the present paper, we focus on the (linear) anisotropic 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. 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 [47]. The spatial domain $(0,1)^d$ can be replaced by any other rectangular domain in $\mathbb{R}^d$ without affecting the essence of this paper. We consider for (1.1) the same space-time approximation as in [9], involving a $\boldsymbol{p}$-degree $C^{\boldsymbol{k}}$ finite element (FE) 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 FE 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 isogeometric analysis (IgA) paradigm [3, 11, 14, 38]. The solver is essentially a preconditioned GMRES (PGMRES) method [48] whose preconditioner P is inspired by the spectral analysis carried out in [9]. From a formal viewpoint, however, it cannot be referred to as a PGMRES because, at each iteration, the linear system associated with P is not solved exactly. Indeed, instead of solving the linear system associated with P, the solver simply applies to it a single iteration of a suitable multigrid method involving no smoothing steps at the finest level, a single Gauss-Seidel smoothing step at coarse levels, and standard bisection for the interpolation and restriction operators at all levels (following the Galerkin approach). From another perspective, we can say that P is approximated at each iteration by some $\tilde{P}$ and we solve exactly the linear system associated with $\tilde{P}$. The solver's performance is illustrated through numerical experiments and turns out to be highly satisfactory in terms of iteration count and run-times. 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 Section 2, we briefly recall the space-time FE-DG discretization of (1.1) and we report the main result of [9] concerning the spectral distribution of the associated discretization matrix C. In Section 3, we present a fast PGMRES method for the matrix C, which is the root from which the proposed solver originated. In Section 4, we describe the proposed solver, and in Section 5, we illustrate its performance in terms of iteration count, run-times and scaling. Finally, in Section 6, we draw conclusions and suggest future lines of research. In order to keep this paper as concise as possible, we borrow notation and terminology from [9]. It is therefore recommended that the reader takes a look at Sections 1 and 2 of [9].

2 Space-Time FE-DG Discretization of Anisotropic Diffusion

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:

$$t_i = i\Delta t, \qquad i = 0, \ldots, N, \qquad \Delta t = \mathrm{T}/N,$$

$$\mathbf{x}_{\boldsymbol{i}} = \boldsymbol{i}\Delta\mathbf{x} = (i_1\Delta x_1, \ldots, i_d \Delta x_d), \qquad \boldsymbol{i} = \mathbf{0}, \ldots, \boldsymbol{n}, \qquad \Delta\mathbf{x} = (\Delta x_1, \ldots, \Delta x_d) = (1/n_1, \ldots, 1/n_d).$$

We consider for the differential problem (1.1) the same space-time discretization as in [9], i.e., we use a $\boldsymbol{p}$-degree $C^{\boldsymbol{k}}$ FE 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 FE basis functions in direction x_i. As carefully explained in [9, Section 3], the overall discretization process leads to the following, normally large, linear system:

$$C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)\mathbf{u} = \mathbf{f}, \tag{2.2}$$

where:

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

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

- 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 $C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$);

- $M_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}$ 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

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

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

  where $B_{\boldsymbol{0},[\boldsymbol{p},\boldsymbol{k}]}, \ldots, B_{\boldsymbol{n(p-k)+k},[\boldsymbol{p},\boldsymbol{k}]}$ are the tensor-product B-splines defined by

$$B_{\boldsymbol{i},[\boldsymbol{p},\boldsymbol{k}]}(\mathbf{x}) = \prod_{r=1}^{d} B_{i_r,[p_r,k_r]}(x_r), \quad \boldsymbol{i} = \boldsymbol{0}, \ldots, \boldsymbol{n(p-k)+k},$$

  with $B_{0,[p_r,k_r]}, \ldots, B_{n_r(p_r-k_r)+k_r,[p_r,k_r]}$ being the B-splines of degree p_r and smoothness C^{k_r} defined on the knot sequence

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

- $M_{[q]}$, $K_{[q]}$, $J_{[q]}$ 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 $\mathbb{P}_q$ of polynomials of degree $\leq q$. In the context of (nodal) DG methods [34], $\ell_{1,[q]}, \ldots, \ell_{q+1,[q]}$ 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 (2.2) yields the approximate solution of problem (1.1); see [9] for details. The main result of [9] is reported in Theorem 2.1 below. Before stating it, 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

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

for all compactly supported functions $F: \mathbb{C}\to\mathbb{C}$. In this case, $\mathbf{f}$ is referred to as the spectral symbol of the sequence $\{X_m\}_m$.

Remark 2.1. The informal meaning behind Definition 2.1 is the following: assuming that $\mathbf{f}$ possesses s Riemann-integrable eigenvalue functions $\lambda_i(\mathbf{f}(\mathbf{y}))$, $i = 1,\dots,s$, 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

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

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

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

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 $\mathbf{0} \leq \boldsymbol{k} \leq \boldsymbol{p}-\mathbf{1}$. Suppose that $K: (0,1)^d \to \mathbb{R}^{d\times d}$ is a symmetric matrix-valued function in $L^\infty((0,1)^d)$ and that the following two conditions are met:*

- *$\boldsymbol{n} = \boldsymbol{\alpha}n$, where $\boldsymbol{\alpha} = (\alpha_1,\dots,\alpha_d)$ 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* [9, *Eq.* (5.12)]*;*
- *T is the final time in* (1.1) *and $M_{[q]}$ is given in* (2.7)*.*

With the same argument used in [9] to prove Theorem 2.1, it is not difficult to prove the following result.

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

$$X_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K) = 2Nn^{d-2}\Big(I_N\otimes\frac{\Delta t}{2}M_{[q]}\otimes K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K)\Big) = \mathrm{T}n^{d-2}I_N\otimes M_{[q]}\otimes K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K).$$

Then,

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

Table 3.1: 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 (2.2), up to a precision $\varepsilon = 10^{-6}$, in the case where $d = 2$, $K(\mathbf{x}) = I_2$, $f(t,\mathbf{x}) = 1$, T = 1, $q = 0$, $\boldsymbol{n} = (n,n)$, $\boldsymbol{p} = (p,p)$, $\boldsymbol{k} = (p-1,p-1)$, $N = n$. The total size of the space-time system (number of DoFs) is given by $n\bar{n} = n(n+p-2)^2$.

$n = N$	GM[2]	PGM[2]	GM[4]	PGM[4]	GM[6]	PGM[6]	GM[8]	PGM[8]
20	43	16	45	16	52	16	82	16
40	104	26	112	26	127	26	152	26
60	179	37	193	37	219	37	242	37
80	266	47	289	47	329	47	365	47
100	363	58	396	58	453	58	508	58
120	470	68	515	68	589	68	660	68

3 Fast PGMRES for the Space-Time FE-DG Matrix $C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$

Suppose the hypotheses of Theorem 2.1 are valid. Then, on the basis of Theorem 2.2 and the theory of (block) generalized locally Toeplitz (GLT) sequences [7, 8, 31, 32, 50, 51], we expect that the sequence of preconditioned matrices

$$(I_N \otimes A_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K))^{-1} C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K), \tag{3.12}$$

as well as the sequence of preconditioned matrices

$$(X_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K))^{-1}(2Nn^{d-2}C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)) = \frac{2}{\Delta t}\,(I_N \otimes M_{[q]} \otimes K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K))^{-1} C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K), \tag{3.13}$$

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 (3.12) and (3.13) are (weakly) clustered at 1; see [7, Section 2.4.2]. Therefore, in view of the convergence properties of the GMRES method [48], we may expect that the PGMRES with preconditioner

$$I_N \otimes A_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K) \tag{3.14}$$

or

$$P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K) = \frac{\Delta t}{2}\, I_N \otimes M_{[q]} \otimes K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K) \tag{3.15}$$

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.

To show that this expectation is realized, we solve the system (2.2) in two space dimensions ($d = 2$), up to a precision $\varepsilon = 10^{-6}$, by means of the GMRES and the PGMRES with preconditioner $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$, using $f(t,\mathbf{x}) = 1$, T = 1, $\boldsymbol{\alpha} = (1,1)$, $\boldsymbol{n} = \boldsymbol{\alpha}n = (n,n)$, $\boldsymbol{p} = (p,p)$, $\boldsymbol{k} = (k,k)$, and varying $K(\mathbf{x})$, N, n, q, p, k. The resulting number of iterations are collected in Tables 3.1–3.3. 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 the spatial parameters n, p, k. However, 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, in order to prove that the proposed PGMRES is fast, we have to show that we are able to solve efficiently a linear system associated with $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$. This is the subject of Sections 3.1 and 3.2. It should be noted, however, that, while the discussion in Section 3.1 is general, in Section 3.2 we need the IgA assumption $\boldsymbol{k} = \boldsymbol{p} - \mathbf{1}$.

3.1 Fast Tensor Solver for the PGMRES Preconditioner $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$

The main observation of this section is that, thanks to the tensor structure of $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$ (see (3.15)), the solution of a linear system with coefficient matrix $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$ reduces to the solution of three linear systems with coefficient

Table 3.2: 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 (2.2), up to a precision $\varepsilon = 10^{-6}$, 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 $= 1$, $q = 1$, $\boldsymbol{n} = (n, n)$, $\boldsymbol{p} = (p, p)$, $\boldsymbol{k} = (k, k)$, $N = 20$. The number of DoFs is given by $40\bar{n} = 40(n(p-k)+k-1)^2$.

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	147	38	231	38	94	38	165	38
40	302	38	469	38	188	38	337	38
60	459	38	707	38	285	38	509	38
80	617	38	945	38	381	38	682	38
100	775	38	1184	38	478	38	856	38
120	933	38	1424	38	575	38	1030	38
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	224	38	238	38	190	38	171	38
40	457	38	325	38	382	38	283	38
60	691	38	463	38	576	38	427	38
80	926	38	623	38	770	38	572	38
100	1161	38	782	38	964	38	718	38
120	1372	38	942	38	1158	38	864	38

Table 3.3: 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 (2.2), up to a precision $\varepsilon = 10^{-6}$, 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 $= 1$, $q = 2$, $\boldsymbol{n} = (n, n)$, $\boldsymbol{p} = (p, p)$, $\boldsymbol{k} = (k, k)$, $N = 20$. The number of DoFs is given by $60\bar{n} = 60(n(p-k)+k-1)^2$.

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	158	24	63	24	220	24	61	24
40	319	24	127	24	442	24	124	24
60	480	24	192	24	665	24	187	24
80	641	24	258	24	889	24	252	24
100	802	24	324	24	1112	24	317	24
120	963	24	390	24	1335	24	382	24
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	350	24	64	24	358	24	71	24
40	573	24	131	24	714	24	140	24
60	859	24	199	24	1070	24	213	24
80	1146	24	268	24	1426	24	287	24
100	1433	24	338	24	1782	24	361	24
120	1720	24	408	24	2138	24	437	24

matrices I_N, $M_{[q]}$, $K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K)$. Indeed, using the canonical algorithm for tensor-product matrices to solve the system $P^{[q,\boldsymbol{p},\boldsymbol{k}]}_{N,\boldsymbol{n}}(K)\mathbf{x}=\mathbf{y}$, we obtain

$$\begin{aligned}
\mathbf{x} &= (P^{[q,\boldsymbol{p},\boldsymbol{k}]}_{N,\boldsymbol{n}}(K))^{-1}\mathbf{y}\\
&= \Big(\frac{2}{\Delta t}I_N \otimes M_{[q]}^{-1} \otimes K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K)^{-1}\Big)\mathbf{y}\\
&= (\widetilde{M}_{N,[q]} \otimes K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K)^{-1})\mathbf{y}\\
&= (\widetilde{M}_{N,[q]} \otimes I_{\bar{n}})(I_{N(q+1)} \otimes K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K)^{-1})\mathbf{y}\\
&= (\widetilde{M}_{N,[q]} \otimes I_{\bar{n}}) \begin{bmatrix} K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K)^{-1} & & & \\ & K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K)^{-1} & & \\ & & \ddots & \\ & & & K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K)^{-1} \end{bmatrix}\mathbf{y} \qquad (3.16)\\
&= \mathrm{vec}(K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K)^{-1}Y\widetilde{M}_{N,[q]}), \qquad (3.17)
\end{aligned}$$

where:

- $\widetilde{M}_{N,[q]} = \frac{2}{\Delta t}I_N \otimes M_{[q]}^{-1}$ can be pre-computed with a negligible cost, because $M_{[q]}$ is a small $(q+1)\times(q+1)$ matrix (if Gauss–Radau nodes are used, $M_{[q]}$ is also diagonal and hence $\widetilde{M}_{N,[q]}$ is diagonal as well);
- $\mathrm{vec}(X)$ is the column-wise form of X, that is the vector obtained by stacking the columns of X;
- Y is the $\bar{n}\times N(q+1)$ matrix such that $\mathrm{vec}(Y)=\mathbf{y}$.

It is then clear that the computation of the solution $\mathbf{x}$ reduces to solving the $N(q+1)$ linear systems $K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K)\mathbf{x}_i=\mathbf{y}_i$, $i=1,\dots,N(q+1)$, where $\mathbf{y}_i$ is the ith column of Y, and multiply the resulting matrix $K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K)^{-1}Y$ by $\widetilde{M}_{N,[q]}$. 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 $i\neq j$. Depending on the implementation and the parallel setting, it can be advantageous to express $\mathbf{x}$ using $\mathrm{vec}(\cdot)$ as in (3.17) or tensor products as in (3.16). The next section is devoted to show how to solve efficiently a linear system associated with $K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{k}]}(K)$ in the maximal smoothness case $\boldsymbol{k}=\boldsymbol{p}-\mathbf{1}$, that is, the case corresponding to IgA. For notational simplicity, the matrix $K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{p}-\mathbf{1}]}(K)$ will be denoted by $K_{\boldsymbol{n},[\boldsymbol{p}]}(K)$.

3.2 Fast Multigrid for the IgA Space Stiffness Matrix $K_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{p}-\mathbf{1}]}(K) = K_{\boldsymbol{n},[\boldsymbol{p}]}(K)$

The fast multigrid solver for $K_{\boldsymbol{n},[\boldsymbol{p}]}(K)$ is inspired by the works [16, 17, 19]. However, we here adopt a slight simplification with respect to [16, 17, 19]. Indeed, while the solver proposed in [16, 17, 19] consists of a preconditioned conjugate gradient (PCG) method with preconditioner $K_{\boldsymbol{n},[\boldsymbol{p}]}(I_d)$ and a fast multigrid solver for $K_{\boldsymbol{n},[\boldsymbol{p}]}(I_d)$, we here simplify the approach and apply directly to $K_{\boldsymbol{n},[\boldsymbol{p}]}(K)$ the fast multigrid solver used in [16, 17, 19] for $K_{\boldsymbol{n},[\boldsymbol{p}]}(I_d)$. This multigrid solver, which was devised on the basis of the spectral symbol derived in [18, 29, 30], can be described as follows.

- At the finest level, we perform ν post-smoothing steps by means of the PCG method. The proposed preconditioner used on the finest level is either $M_{\boldsymbol{n},[\boldsymbol{p},\boldsymbol{p}-\mathbf{1}]} = M_{\boldsymbol{n},[\boldsymbol{p}]}$ (i.e., the mass matrix) or $M_{\boldsymbol{n}+\mathbf{1},[\boldsymbol{p}-\mathbf{1},\boldsymbol{p}-\mathbf{1}]} = M_{\boldsymbol{n}+\mathbf{1},[\boldsymbol{p}-\mathbf{1}]}$ (i.e., the mass matrix arising from the $(\boldsymbol{p}-\mathbf{1})$-degree discretization, with a correction from $\boldsymbol{n}$ to $\boldsymbol{n}+\mathbf{1}$ in order to meet the size of $K_{\boldsymbol{n},[\boldsymbol{p}]}(K)$) or the natural Toeplitz modification $\overline{M}_{\boldsymbol{n}+\mathbf{1},[\boldsymbol{p}-\mathbf{1}]}$ of $M_{\boldsymbol{n}+\mathbf{1},[\boldsymbol{p}-\mathbf{1}]}$. In order to explain what we mean by "natural Toeplitz modification" of $M_{\boldsymbol{n}+\mathbf{1},[\boldsymbol{p}-\mathbf{1}]}$, let us recall from [9, Lemmas 4.1 and 4.2] that

$$M_{\boldsymbol{n}+\mathbf{1},[\boldsymbol{p}-\mathbf{1}]} = \bigotimes_{r=1}^{d} M_{n_r+1,[p_r-1]},$$

where $M_{n_r+1,[p_r-1]}$ is a banded Toeplitz matrix, up to a small-rank correction whose nonzero entries are gathered at the upper-left and lower-right corners. Denoted by $\overline{M}_{n_r+1,[p_r-1]}$ the Toeplitz matrix obtained from $M_{n_r+1,[p_r-1]}$ by removing the small-rank correction, the matrix $\overline{M}_{\boldsymbol{n}+\mathbf{1},[\boldsymbol{p}-\mathbf{1}]}$ is defined as

$$\overline{M}_{\boldsymbol{n}+\mathbf{1},[\boldsymbol{p}-\mathbf{1}]} = \bigotimes_{r=1}^{d} \overline{M}_{n_r+1,[p_r-1]},$$

Table 3.4: Number of iterations needed by a 5-level cascade multigrid for solving the linear system $K_{\boldsymbol{n},[\boldsymbol{p}]}(K)\mathbf{y} = \mathbf{1}$ up to a precision $\varepsilon = 10^{-8}$, in the case where $d = 2$, $K(\mathbf{x}) = I_2$, $\boldsymbol{n} = (n, n)$, $\boldsymbol{p} = (p, p)$. We perform ν smoothing steps at the finest level and a single Gauss-Seidel iteration at coarse levels. The smoother at the finest level is explicitly indicated in the table. We use $n = 259 - p$, corresponding to a number of DoFs equal to $\overline{n} = 257^2$.

$p\,[\nu]$	2 [3]	3 [4]	4 [4]	5 [5]	6 [6]	7 [7]	8 [8]
Gauss-Seidel smoother	5	5	5	22	114	384	2154
ILU($K_{\boldsymbol{n},[\boldsymbol{p}]}(K)$) PCG smoother	5	5	5	7	20	68	282
$M_{\boldsymbol{n},[\boldsymbol{p}]}$ PCG smoother	6	6	7	7	7	8	9
$M_{\boldsymbol{n+1},[\boldsymbol{p-1}]}$ PCG smoother	6	5	6	6	7	9	12
$\overline{M}_{\boldsymbol{n+1},[\boldsymbol{p-1}]}$ PCG smoother	6	5	5	6	6	7	12

Table 3.5: Number of iterations needed by a 5-level cascade multigrid for solving the linear system $K_{\boldsymbol{n},[\boldsymbol{p}]}(K)\mathbf{y} = \mathbf{1}$ up to a precision $\varepsilon = 10^{-8}$, in the case where $d = 2$,

$$K(x_1, x_2) = \begin{bmatrix} (3 + x_1)e^{x_2} & 0.1 \\ -0.1(x_1 + x_2) & \cos x_1 + 2 \end{bmatrix},$$

$\boldsymbol{n} = (n, n)$, $\boldsymbol{p} = (p, p)$. We perform ν smoothing steps at the finest level and a single Gauss-Seidel iteration at coarse levels. The smoother at the finest level is explicitly indicated in the table. We use $n = 259 - p$, corresponding to a number of DoFs equal to $\overline{n} = 257^2$. Experiments are run in parallel on 20 processors.

$p\,[\nu]$	2 [3]	3 [4]	4 [4]	5 [5]	6 [6]	7 [7]	8 [8]
Gauss-Seidel smoother	8	6	15	31	120	395	2260
ILU($K_{\boldsymbol{n},[\boldsymbol{p}]}(K)$) PCG smoother	7	6	6	10	25	71	348
$M_{\boldsymbol{n},[\boldsymbol{p}]}$ PCG smoother	8	9	13	13	15	17	19
$M_{\boldsymbol{n+1},[\boldsymbol{p-1}]}$ PCG smoother	8	8	9	10	12	18	28
$\overline{M}_{\boldsymbol{n+1},[\boldsymbol{p-1}]}$ PCG smoother	8	8	7	7	9	12	23

and it is again a Toeplitz matrix or, more precisely, a d-level Toeplitz matrix; see [32, Lemma 3.3]. We note that, due to the tensor structure of the three matrices $M_{\boldsymbol{n},[\boldsymbol{p}]}$, $M_{\boldsymbol{n+1},[\boldsymbol{p-1}]}$, $\overline{M}_{\boldsymbol{n+1},[\boldsymbol{p-1}]}$, a linear system associated with any of these matrices can be solved by the usual tensor solver already described above for $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{k}]}(K)$; this solver essentially reduces the solution of the given system to the solution of d systems associated with the single factors (that is, $M_{n_r,[p_r]}$, $r = 1, \ldots, d$, in the case of $M_{\boldsymbol{n},[\boldsymbol{p}]}$, $M_{n_r+1,[p_r-1]}$, $r = 1, \ldots, d$, in the case of $M_{\boldsymbol{n+1},[\boldsymbol{p-1}]}$, and $\overline{M}_{n_r+1,[p_r-1]}$, $r = 1, \ldots, d$, in the case of $\overline{M}_{\boldsymbol{n+1},[\boldsymbol{p-1}]}$).

- At all coarse levels, we use one post-smoothing iteration by the Gauss-Seidel method.
- At all levels, we use standard bisection for the interpolation and restriction operators, and we assemble coarse-grid operators through Galerkin assembly.

We remark that the proposed solver is a cascade multigrid as the pre-smoothing is absent at each level.

In Tables 3.4 and 3.5, we compare the multigrid convergence using different smoothers at the finest level and varying p. We note how the multigrid performance with respect to p differs using the listed approaches. In particular, for standard Gauss-Seidel smoothing, the multigrid convergence quickly deteriorates when increasing p. This is actually expected, because, when using the Gauss-Seidel smoother, the resulting solver is a standard multigrid algorithm, whose convergence rate is known to worsen rapidly with increasing p; see [16, 17, 19, 24, 25].

Remark 3.1. Clearly, it is worth preconditioning $K_{\boldsymbol{n},[\boldsymbol{p}]}(K)$ with $M_{\boldsymbol{n},[\boldsymbol{p}]}$ (resp., $M_{\boldsymbol{n+1},[\boldsymbol{p-1}]}$, $\overline{M}_{\boldsymbol{n+1},[\boldsymbol{p-1}]}$) only if the latter matrix can be solved efficiently. This is the case in our context where the three matrices $M_{\boldsymbol{n},[\boldsymbol{p}]}$, $M_{\boldsymbol{n+1},[\boldsymbol{p-1}]}$, $\overline{M}_{\boldsymbol{n+1},[\boldsymbol{p-1}]}$ possess a tensor structure, due to the rectangular nature of the physical domain. In practice, using as preconditioner the incomplete LU (ILU) factorization of $K_{\boldsymbol{n},[\boldsymbol{p}]}(K)$, like in the second line of Tables 3.4 and 3.5, can be competitive in terms of run-time and can be adopted when dealing with unstructured spatial grids, where the tensor structure of $M_{\boldsymbol{n},[\boldsymbol{p}]}$, $M_{\boldsymbol{n+1},[\boldsymbol{p-1}]}$, $\overline{M}_{\boldsymbol{n+1},[\boldsymbol{p-1}]}$ is no longer available.

4 Fast Solver for the Space-Time IgA-DG Matrix $C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{p-1}]}(K) = C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)$

The solver suggested in Section 3 for a linear system with coefficient matrix $C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{p-1}]}(K) = C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)$ is a PGMRES with preconditioner $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p},\boldsymbol{p-1}]}(K) = P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)$; the solution of a linear system associated with $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)$, which is required at each PGMRES iteration, is performed via the tensor solver described in Section 3.1 coupled with the multigrid method described in Section 3.2 for the space stiffness matrix $K_{\boldsymbol{n},[\boldsymbol{p}]}(K)$.

Actually, it was discovered experimentally that *the PGMRES method converges faster if the linear system with coefficient matrix $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)$ occurring at each PGMRES iteration is not solved exactly.* More precisely, when applying to $K_{\boldsymbol{n},[\boldsymbol{p}]}(K)$ the multigrid method described in Section 3.2, *it is enough to perform only a few multigrid iterations in order to achieve an excellent PGMRES run-time and, in fact, only one multigrid iteration is sufficient.* Even more astonishing, it was discovered experimentally that *the PGMRES run-time is further improved if we do not perform any smoothing step at the finest level in the few multigrid iterations applied to $K_{\boldsymbol{n},[\boldsymbol{p}]}(K)$* (that is, smoothing steps occur only at coarse levels).

In view of these experimental discoveries, we propose to solve a linear system with coefficient matrix $C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)$ in the following way:

- apply to the given system the PGMRES algorithm with preconditioner $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)$;
- apply to the linear system with coefficient matrix $P_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)$ occurring at each PGMRES iteration the tensor solver described in Section 3.1;
- the tensor solver would require solving $q(N+1)$ linear systems with coefficient matrix $K_{\boldsymbol{n},[\boldsymbol{p}]}(K)$ as per Eq. (3.16) or (3.17); instead of solving exactly these systems, apply to each of them μ multigrid iterations involving no smoothing steps at the finest level, a single Gauss-Seidel post-smoothing step at coarse levels, and standard bisection for the interpolation and restriction operators at all levels (following the Galerkin approach).

Note that this solver cannot be referred to as a PGMRES because, at each iteration, the linear system associated with the preconditioner is not solved exactly. As we shall see in the numerics of Section 5, the choice $\mu = 1$ yields the best performance of the solver.

5 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 solvers, such as the PGMRES with ILU preconditioner.

5.1 Implementation Details

For the numerics of this section, as well as throughout this paper, we used the C++ frameworks PETSc [5, 6] and the domain specific language Utopia [60] for the parallel linear algebra and solvers. We stress that, when multiple processors are used, a block Jacobi preconditioning is employed by default by PETSc. For the assembly of high order finite elements, we used the PetIGA package [15]. A parallel tensor-product routine had to implemented to assemble space-time matrices. Numerical experiments have been performed on the Piz Daint supercomputer of the Swiss national supercomputing centre (CSCS).[1]

5.2 Experimental Setting

In the numerics of this section, we solve the linear system (2.2) 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 $\ell_{1,[q]}, \dots, \ell_{q+1,[q]}$ 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 $\varepsilon = 10^{-8}$ as a tolerance and the PETSc default stopping criterion. 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 brackets next to the corresponding run-time. Throughout this section, we use the following abbreviations for the solvers.

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

Table 5.6: PGMRES iterations and run-time (using 64 cores) to solve the linear system (2.2) up to a precision of 10^{-8}, according to the experimental setting described in Section 5.2. We used $K(\mathbf{x}) = I_2$, $q = 0$, $N = 32$ time steps and $n = 259 - p$. The total size of the space-time system (number of DoFs) is given by $32 \cdot 257^2$.

p	1	2	3	4	5	6	7	8	9
ILU($C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)$) PGMRES	6.0 [545]	7.5 [332]	7.7 [249]	8.0 [211]	8.5 [182]	10.1 [164]	11.0 [147]	15.7 [166]	20.3 [174]
$\text{MG}_{3,2}^5$ PGMRES	1.5 [33]	3.0 [33]	5.2 [33]	7.7 [33]	11.0 [34]	16.3 [36]	21.1 [38]	30.5 [43]	42.4 [50]
$\text{MG}_{1,2}^5$ PGMRES	0.9 [33]	1.7 [33]	2.8 [33]	4.2 [34]	6.4 [37]	9.7 [41]	13.6 [44]	19.5 [52]	27.6 [61]
$\text{MG}_{3,0}^5$ PGMRES	1.1 [33]	2.0 [33]	2.5 [33]	3.5 [33]	4.5 [33]	6.1 [33]	7.6 [33]	9.5 [33]	11.4 [33]
$\text{MG}_{1,0}^5$ PGMRES	0.7 [33]	1.2 [33]	1.4 [33]	2.0 [33]	2.6 [33]	3.4 [33]	4.2 [33]	5.4 [33]	6.5 [33]

Table 5.7: PGMRES iterations and run-time (using 64 cores) to solve the linear system (2.2) up to a precision of 10^{-8}, according to the experimental setting described in Section 5.2. We used

$$K(x_1, x_2) = \begin{bmatrix} \cos(x_1) + x_2 & 0 \\ 0 & x_1 + \sin(x_2) \end{bmatrix},$$

$q = 1$, $N = 20$ time steps and $n = 131 - p$. The total size of the space-time system (number of DoFs) is given by $40 \cdot 129^2$.

p	1	2	3	4	5	6	7	8	9
ILU($C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)$) PGMRES	1.3 [442]	2.0 [282]	2.7 [210]	3.0 [178]	3.5 [153]	4.0 [137]	5.0 [132]	6.9 [143]	10.4 [176]
$\text{MG}_{1,0}^5$ PGMRES	0.4 [46]	0.7 [46]	0.9 [46]	1.2 [45]	1.6 [45]	2.1 [46]	2.7 [47]	3.3 [47]	4.2 [49]

- **ILU($C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)$) PGMRES**

  PGMRES with preconditioner given by an ILU factorization of the system matrix $C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}$.

- **$\text{MG}_{\mu,\nu}^L$ PGMRES**

  The proposed solver, as described in Section 4, with μ multigrid iterations applied to $K_{\boldsymbol{n},[\boldsymbol{p}]}(K)$. Each multigrid iterations involves ν Gauss-Seidel smoothing steps at the finest level (typically $\nu = 0$) and 1 Gauss-Seidel smoothing step at the coarse levels. The superscript L denotes the number of multigrid levels.

- **$\text{TMG}_{\mu,\nu}^L$ PGMRES**

  The same as "$\text{MG}_{\mu,\nu}^L$ PGMRES", with the only difference that the multigrid iterations are performed with the telescopic option, thus giving rise to the telescopic multigrid (TMG) [20, 44]. This technique consists in halving the number of processors used across the grid hierarchy: if N_f processors are used on the fine grid ($l = 0$), then we use $N_f/2^l$ processors on level l. This strategy can be beneficial for the parallel multigrid performance, as shown in Section 5.4.

5.3 Iteration Count and Timing

Tables 5.6–5.8 illustrate the performance of the proposed solver in terms of number of iterations and run-times. It is clear from the table that the solver is superior to the classical PGMRES with preconditioner given by the ILU factorization of the system matrix $C_{N,\boldsymbol{n}}^{[q,\boldsymbol{p}]}(K)$. Moreover, the best performance of the solver is obtained when applying to $K_{\boldsymbol{n},[\boldsymbol{p}]}(K)$ a single multigrid iteration ($\mu = 1$) with no smoothing steps at the finest level ($\nu = 0$). It should also be noted that the solver is considerably robust with respect to the spline degree p as both number of iterations and run-time do not grow significantly with p.

5.4 Scaling

In the scaling experiments, besides the multigrid already considered above, we also use a TMG for performance reasons (see Section 5.2 for some details/references about the TMG). From Table 5.9 and Figure 5.1 we see that the proposed

Table 5.8: PGMRES iterations and run-time (using 64 cores) to solve the linear system (2.2) up to a precision of 10^{-8}, according to the experimental setting described in Section 5.2. We used

$$K(x_1, x_2) = \begin{bmatrix} (2+\cos x_1)(1+x_2) & \cos(x_1+x_2)\sin(x_1+x_2) \\ \cos(x_1+x_2)\sin(x_1+x_2) & (2+\sin x_2)(1+x_1) \end{bmatrix},$$

$q = 0$, $N = 20$ time steps and $n = 259 - p$. The total size of the space-time system (number of DoFs) is given by $20 \cdot 257^2$.

p	1	2	3	4	5	6	7	8	9
ILU($C^{[q,\boldsymbol{p}]}_{N,\boldsymbol{n}}(K)$) PGMRES	2.2 [441]	2.7 [256]	3.4 [189]	3.7 [161]	4.2 [141]	4.9 [127]	5.7 [115]	9.5 [151]	41.2 [477]
$\mathrm{MG}^5_{1,0}$ PGMRES	0.2 [12]	0.4 [11]	0.4 [11]	0.6 [11]	0.7 [12]	0.9 [12]	1.2 [13]	2.4 [13]	1.9 [14]

Table 5.9: Strong scaling: PGMRES iterations and run-time to solve the linear system (2.2) up to a precision of 10^{-8}, according to the experimental setting described in Section 5.2. We used $K(\mathbf{x}) = I_2$, $q = 0$, $p = 3$, $N = 64$ time steps and $n = 384$. The total size of the space-time system (number of DoFs) is given by $64 \cdot 385^2$.

Cores	1	2	4	8	16	32	
ILU($C^{[q,\boldsymbol{p}]}_{N,\boldsymbol{n}}(K)$) PGMRES	2591 [394]	1220 [403]	603 [394]	340.9 [396]	232.6 [396]	112.0 [397]	
$\mathrm{MG}^7_{1,0}$ PGMRES	304.2 [63]	139.4 [63]	70.0 [63]	39.7 [63]	20.4 [63]	26.1 [63]	
$\mathrm{TMG}^7_{1,0}$ PGMRES	304.2 [63]	142.5 [63]	69.5 [63]	41.5 [63]	25.9 [63]	21.3 [63]	
Cores	64	128	256	512	1024	2048	4096
ILU($C^{[q,\boldsymbol{p}]}_{N,\boldsymbol{n}}(K)$) PGMRES	99.3 [399]	55.0 [479]	28.7 [497]	18.1 [531]	11.5 [603]	60 [2397]	35.3 [2571]
$\mathrm{MG}^7_{1,0}$ PGMRES	12.0 [63]	6.2 [64]	3.4 [64]	2.6 [64]	4.7 [65]	16.9 [65]	65.0 [65]
$\mathrm{TMG}^7_{1,0}$ PGMRES	10.7 [63]	5.7 [63]	2.8 [62]	1.7 [62]	1.0 [62]	0.85 [62]	0.83 [62]

solver, especially when using the TMG option, shows a nearly optimal strong scaling with respect to the number of cores. Table 5.10 and Figure 5.2 illustrate the weak scaling properties of the proposed solver, which possesses a remarkably superior parallel efficiency with respect to the standard ILU approach in terms of iteration count and run-time. In fact, the efficiency of the proposed solver can be estimated to be about three times the one of the standard ILU approach.

6 Conclusions and Perspectives

We have proposed a new solver for the space-time IgA-DG discretization of the anisotropic diffusion problem (1.1). Through numerical experiments, we have illustrated the competitiveness of our proposal with respect to other benchmark solvers in terms of iteration count, run-times and scaling. In particular, the solver is suited for parallel computation as it shows remarkable scalability properties with respect to the number of cores.

We conclude this work by suggesting a few possible future lines of research/investigation.

- Provide a theoretical analysis of the proposed solver, possibly giving a formal proof of its convergence properties.
- Investigate the performance of the solver for the anisotropic diffusion problem (1.1) in the case of a space domain Ω more complex than the hypersquare $(0,1)^d$.

Table 5.10: Space-time weak scaling: PGMRES iterations and run-time to solve the linear system (2.2) up to a precision of 10^{-8}, according to the experimental setting described in Section 5.2. 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.

[Cores, n, N, L]	[1, 65, 8, 4]	[8, 129, 16, 5]	[64, 257, 32, 6]	[512, 513, 64, 7]
ILU($C^{[q,\boldsymbol{p}]}_{N,\boldsymbol{n}}(K)$) PGMRES	0.18 [47]	0.74 [113]	10.9 [332]	103.8 [1354]
$\mathrm{TMG}^L_{1,0}$ PGMRES	0.08 [10]	0.21 [17]	0.90 [33]	1.97 [62]

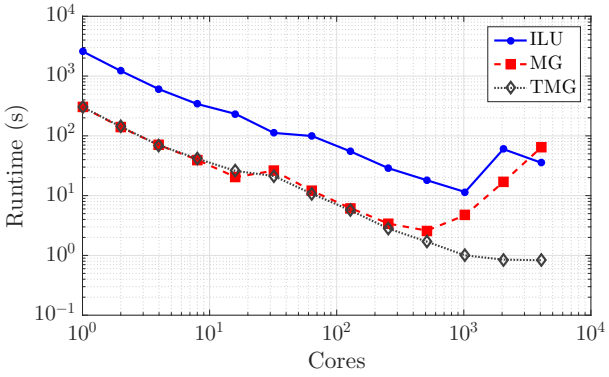

Figure 5.1: Graphical representation of the run-times reported in Table 5.9.

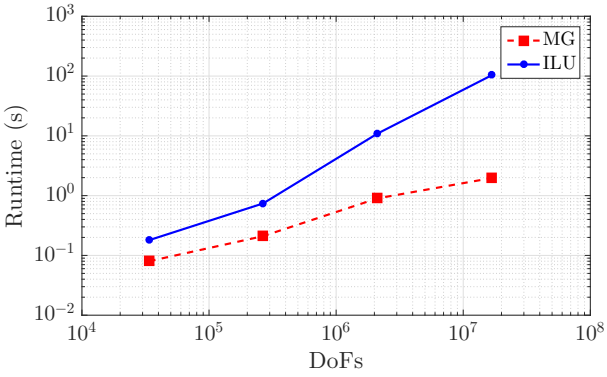

Figure 5.2: Graphical representation of the run-times reported in Table 5.10.

- Consider an inner/outer multilevel hierarchy in time to improve the overall scalability of the proposed solver; for example, using it as a smoother in a multigrid-in-time algorithm.
- Apply the (simple) idea behind the proposed solver in the context of other PDE discretizations. In this regard, it is worth stressing that the proposed solver is highly flexible as it does not depend on the domain or the space-time discretization, as long as a tensor-product structure is maintained between space and time.

Acknowledgements

Carlo Garoni acknowledges the MIUR Excellence Department Project awarded to the Department of Mathematics, University of Rome "Tor Vergata", CUP E83C18000100006. Paola Ferrari is (partially) financed by the GNCS 2019 Project "Metodi Numerici per Problemi Mal Posti". Paola Ferrari, Carlo Garoni and Stefano Serra-Capizzano thank the Italian INdAM-GNCS for the scientific support. 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.

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 (2006) 3247–3273.

[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 (2010) 2075–2107.

[4] Aziz A. K., Monk P. *Continuous finite elements in space and time for the heat equation.* Math. Comp. 52 (1989) 255–274.

[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.* Technical Report 2019-004, Uppsala University (2019).

[8] Barbarino G., Garoni C., Serra-Capizzano S. *Block generalized locally Toeplitz sequences: theory and applications in the multidimensional case.* Technical Report 2019-005, Uppsala University (2019).

[9] 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 (2018) 1383–1420.

[10] 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.

[11] Beirão da Veiga L., Buffa A., Sangalli G., Vázquez R. *Mathematical analysis of variational isogeometric methods.* Acta Numerica 23 (2014) 157–287.

[12] 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 (2001) 1931–1955.

[13] 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 (2012) 1181–1206.

[14] Cottrell J. A., Hughes T. J. R., Bazilevs Y. *Isogeometric Analysis: Toward Integration of CAD and FEA.* John Wiley & Sons, Chicester (2009).

[15] 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 (2016) 151–181.

[16] 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 (2015) 230–264.

[17] 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 (2015) 1120–1146.

[18] Donatelli M., Garoni C., Manni C., Serra-Capizzano S., Speleers H. *Spectral analysis and spectral symbol of matrices in isogeometric collocation methods.* Math. Comp. 85 (2016) 1639–1680.

[19] 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 (2017) 31–62.

[20] Douglas C. C. *A review of numerous parallel multigrid methods.* In "Applications on Advanced Architecture Computers", SIAM (1996), pp. 187–202.

[21] Eriksson K., Johnson C., Logg A. *Adaptive computational methods for parabolic problems: Part 1. Fundamentals.* Encyclop. Comput. Mech. (2004).

[22] Falgout R. D., Friedhoff S., Kolev Tz. V., MacLachlan S. P., Schroder J. B., Vandewalle S. *Multigrid methods with space-time concurrency.* Comput. Visual. Sci. 18 (2007) 123–143.

[23] French D. A. *A space-time finite element method for the wave equation.* Comput. Methods Appl. Mech. Engrg. 107 (1993) 145–157.

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

[25] Gahalaut K. P. S., Tomar S. K., Kraus J. K. *Algebraic multilevel preconditioning in isogeometric analysis: construction and numerical studies.* Comput. Methods Appl. Mech. Engrg. 266 (2013) 40–56.

[26] Gander M. J., Halpern L. *Techniques for locally adaptive time stepping developed over the last two decades.* Domain Decomp. Methods Sci. Engrg. XX (2013) 377–385.

[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., Neumüller M. *Analysis of a new space-time parallel multigrid algorithm for parabolic problems.* SIAM J. Sci. Comput. 38 (2016) A2173–A2208.

[29] Garoni C., Manni C., Pelosi F., Serra-Capizzano S., Speleers H. *On the spectrum of stiffness matrices arising from isogeometric analysis.* Numer. Math. 127 (2014) 751–799.

[30] Garoni C., Manni C., Serra-Capizzano S., Sesana D., Speleers H. *Spectral analysis and spectral symbol of matrices in isogeometric Galerkin methods.* Math. Comp. 86 (2017) 1343–1373.

[31] Garoni C., Serra-Capizzano S. *Generalized Locally Toeplitz Sequences: Theory and Applications.* Vol. I, Springer, Cham (2017).

[32] Garoni C., Serra-Capizzano S. *Generalized Locally Toeplitz Sequences: Theory and Applications.* Vol. II, Springer, Cham (2018).

[33] Griebel M., Oeltz D. *A sparse grid space-time discretization scheme for parabolic problems.* Computing 81 (2007) 1–34.

[34] Hesthaven J. S., Warburton T. *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications.* Springer, New York (2008).

[35] Hofer C., Langer U., Neumüller M. *Parallel and robust preconditioning for space-time isogeometric analysis of parabolic evolution problems.* SIAM J. Sci. Comput. 41 (2019) A1793–A1821.

[36] Horton G., Vandewalle S. *A space-time multigrid method for parabolic partial differential equations.* SIAM J. Sci. Comput. 16 (1995) 848–864.

[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 (1988) 339–363.

[38] 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 (2005) 4135–4195.

[39] 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 (2006) 589–611.

[40] 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 (2016) 164–179.

[41] Ladyženskaja O. A., Solonnikov V. A., Ural'ceva N. N. *Linear and Quasi-Linear Equations of Parabolic Type.* Amer. Math. Soc. (1968).

[42] Langer U., Moore S. E., Neumüller M. *Space-time isogeometric analysis of parabolic evolution problems.* Comput. Methods Appl. Mech. Engrg. 306 (2016) 342–363.

[43] Miller S. T., Haber R. B. *A spacetime discontinuous Galerkin method for hyperbolic heat conduction.* Comput. Methods Appl. Mech. Engrg. 198 (2008) 194–209.

[44] 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).

[45] Meidner D., Vexler B. *Adaptive space-time finite element methods for parabolic optimization problems.* SIAM J. Control Optim. 46 (2007) 116–142.

[46] Neumüller M., Steinbach O. *Refinement of flexible space-time finite element meshes and discontinuous Galerkin methods.* Comput. Visual Sci. 14 (2011) 189–205.

[47] Quarteroni A. *Numerical Models for Differential Problems.* Springer, Milan (2009).

[48] Saad Y. *Iterative Methods for Sparse Linear Systems.* 2nd ed., SIAM, Philadelphia (2003).

[49] Schötzau D., Schwab C. *An hp a priori error analysis of the DG time-stepping method for initial value problems.* Calcolo 37 (2000) 207–232.

[50] Serra-Capizzano S. *Generalized locally Toeplitz sequences: spectral analysis and applications to discretized partial differential equations.* Linear Algebra Appl. 366 (2003) 371–402.

[51] Serra-Capizzano S. *The GLT class as a generalized Fourier analysis and applications.* Linear Algebra Appl. 419 (2006) 180–233.

[52] 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 (1991) 141–219.

[53] 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 (2006) 1491–1518.

[54] 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 (1992) 339–351.

[55] 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 (1992) 353–371.

[56] 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 (2006) 2002–2027.

[57] Thite S. *Adaptive spacetime meshing for discontinuous Galerkin methods.* Comput. Geom. 42 (2009) 20–44.

[58] Thomée V. *Galerkin Finite Element Methods for Parabolic Problems.* Springer, New York (2006).

[59] 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 (2002) 546–585.

[60] 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).