Research Article

Combining *p*-multigrid and Multigrid Reduction in Time methods to obtain a scalable solver for Isogeometric Analysis

Roel Tielen[1] · **Matthias Möller**[1] · **Cornelis Vuik**[1]

Received: 25 November 2021 / Accepted: 22 April 2022
Published online: 10 May 2022

Abstract
The use of sequential time integration schemes becomes more and more the bottleneck within large-scale computations due to a stagnation of processor's clock speeds. In this study, we combine the parallel-in-time Multigrid Reduction in Time method with a *p*-multigrid method to obtain a scalable solver specifically designed for Isogeometric Analysis. Numerical results obtained for two- and three-dimensional benchmark problems show the overall scalability of the proposed method on modern computer architectures and a significant improvement in terms of CPU timings compared to the use of standard spatial solvers.

Article Highlights

- The use of a p-multigrid method significantly reduces the CPU timings for higher values of the spline degree.
- The Multigrid Reduction in Time method shows both strong and weak scalability up to 2048 cores.
- Iteration numbers are independent of the number of time steps, mesh width and spline degree.

Keywords Multigrid Reduction in Time · Isogeometric Analysis · *p*-multigrid

1 Introduction

Since its introduction in [1], Isogeometric Analysis (IgA) has become more and more a viable alternative to the Finite Element Method (FEM). Within IgA, the same building blocks (i.e., B-splines and NURBS) as in Computer Aided Design (CAD) are adopted, which closes the gap between CAD and FEM. In particular, the use of high-order splines results in a highly accurate represention of (curved) geometries and has shown to be advantageous in many applications, like structural mechanics [2], solid and fluid dynamics [3] and shape optimization [4]. Finally, the accuracy per degree of freedom (DOF) compared to FEM is significantly higher with IgA [5].

For time-dependent partial differential equations (PDEs), IgA is typically combined with a traditional time integration scheme within the method of lines. Here, the spatial variables are discretized by adopting IgA, after which the resulting system of ordinary differential equations (ODEs) is integrated in time. However, as with all traditional time integration schemes, the latter part becomes more and more the bottleneck in numerical simulations. When the spatial

Roel Tielen, Matthias Möller, and Cornelis Vuik have contributed equally to this work

✉ Roel Tielen, r.p.w.m.tielen@tudelft.nl; Matthias Möller, m.moller@tudelft.nl; Cornelis Vuik, c.vuik@tudelft.nl | [1]Delft Institute of Applied Mathematics, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands.

resolution is increased to improve accuracy, a smaller time step size has to be chosen to ensure stability of the overall method. As clock speeds are no longer increasing, but the core count goes up, the parallelizability of the entire calculation process becomes more and more important to obtain an overall efficient method. As traditional time integration schemes are sequential by nature, new parallel-in-time methods are needed to resolve this problem.

The Multigrid Reduction in Time (MGRIT) method [6] is a parallel-in-time algorithm based on multigrid reduction (MGR) techniques [7]. In contrast to space-time methods, in which time is considered as an extra spatial dimension, sequential time stepping is still necessary within MGRIT. Space-time methods have been combined in the literature with IgA [8]. Although very successful, a drawback of such methods is the fact that they are more intrusive on existing codes, while MGRIT just requires a routine to integrate the fully discrete problem from one time instance to the next. Over the years, MGRIT has been studied in detail (see [9]) and applied to a variety of problems, including those arising in optimization [10] and power networks [11].

Recently, the authors applied MGRIT in the context of IgA for the first time in the literature [12]. Here, MGRIT showed convergence for a variety of two-dimensional benchmark problems independent of the mesh width h, the spline degree p of the B-spline basis functions and the number of time steps N_t. However, as a standard (diagonally preconditioned) Conjugate Gradient method was adopted for the spatial solves within MGRIT, a significant dependency of the CPU timings on the spline degree was visible. Furthermore, the parallel performance of MGRIT was investigated for a very limited number of cores.

In this paper, we extend the research direction set out in [12] by combining MGRIT with a state-of-the-art p-multigrid method [13] to solve the linear systems arising within MGRIT. CPU timings show that the use of such a solver significantly improves the overall performance of MGRIT, in particular for higher values of p. Furthermore, the parallel performance of the resulting MGRIT method (i.e., strong and weak scalability) is investigated on modern computer architectures, showing significant (and close to ideal) speed-ups up to 2048 cores.

This paper is structured as follows: In Sect. 2, a two-dimensional model problem and its spatial and temporal discretization are considered. The MGRIT algorithm is then described in Sect. 3. In Sect. 4, the adopted p-multigrid method and its components are presented in more detail. In Sect. 5, numerical results obtained for the considered model problem are analyzed for different values of the mesh width, spline degree and the number of time steps and compared to those obtained in [12]. Furthermore, weak and strong scaling studies of MGRIT when adopting a p-multigrid method are performed. Finally, conclusions are drawn in Sect. 7.

2 Model problem and discretization

As a model problem, we consider the transient diffusion equation:

$$\partial_t u(\mathbf{x},t) - \Delta u(\mathbf{x},t) = f(\mathbf{x}), \quad \mathbf{x} \in \Omega, t \in [0,T]. \tag{1}$$

Here, $\Omega \subset \mathbb{R}^d$ denotes a simply connected, Lipschitz domain in d dimensions and $f \in L^2(\Omega)$ a source term. The above equation is complemented by initial conditions and homogeneous Dirichlet boundary conditions:

$$u(\mathbf{x},0) = u^0(\mathbf{x}), \quad \mathbf{x} \in \Omega, \tag{2}$$

$$u(\mathbf{x},t) = 0, \qquad \mathbf{x} \in \partial\Omega, t \in [0,T]. \tag{3}$$

First, we discretize Eq. (1) by dividing the time interval $[0,T]$ in N_t subintervals of size Δt and applying the θ-scheme to the temporal derivative, which leads to the following equation to be solved at every time step:

$$\frac{u(\mathbf{x})^{k+1} - u(\mathbf{x})^k}{\Delta t} = \theta \Delta u(\mathbf{x})^{k+1} + (1-\theta)\Delta u(\mathbf{x})^k + f(\mathbf{x}), \tag{4}$$

for $\mathbf{x} \in \Omega$ and $k = 0, \ldots, N_t$. Depending on the choice of θ, this scheme leads to the backward Euler ($\theta = 1$), forward Euler ($\theta = 0$) or second-order accurate Crank–Nicolson ($\theta = 0.5$) method. By rearranging the terms, the discretized equation can be written as follows:

$$\begin{aligned} &u(\mathbf{x})^{k+1} - \Delta t\theta \Delta u(\mathbf{x})^{k+1} \\ &\quad = u(\mathbf{x})^k + \Delta t(1-\theta)\Delta u(\mathbf{x})^k + \Delta t f(\mathbf{x}). \end{aligned} \tag{5}$$

To obtain the variational formulation, let $\mathcal{V} = H_0^1(\Omega)$ be the space of functions in the Sobolev space $H^1(\Omega)$ that vanish on the boundary $\partial\Omega$. Equation (5) is multiplied with a test function $v \in \mathcal{V}$ and the result is then integrated over the domain Ω:

$$\begin{aligned} &\int_\Omega u^{k+1}v - \Delta t\theta \Delta u^{k+1} v \,\mathrm{d}\Omega \\ &\quad = \int_\Omega u^k v + \Delta t(1-\theta)\Delta u^k v + \Delta t f v \,\mathrm{d}\Omega, \end{aligned} \tag{6}$$

where we write $u(\mathbf{x})^{k+1} = u^{k+1}$ throughout the remainder of this section to improve readability. Applying integration by parts on the second term on both sides of the equation results in

$$\begin{aligned} &\int_\Omega u^{k+1}v + \Delta t\theta \nabla u^{k+1} \cdot \nabla v \,\mathrm{d}\Omega \\ &\quad = \int_\Omega u^{k+1}v - \Delta t(1-\theta)\nabla u^k \cdot \nabla v + \Delta t f v \,\mathrm{d}\Omega, \end{aligned} \tag{7}$$

where the boundary integral integral vanishes since $v = 0$ on $\partial\Omega$. To parameterize the physical domain Ω, a geometry

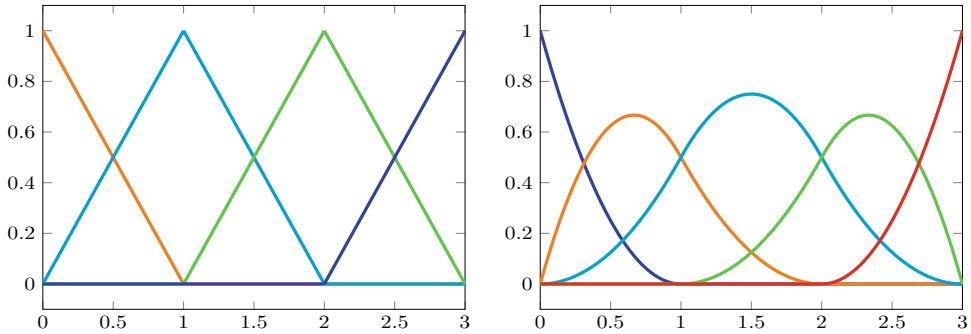

Fig. 1 Linear (left) and quadratic (right) B-spline basis functions based on the knot vectors $\Xi_1 = \{0, 0, 1, 2, 3, 3\}$ and $\Xi_2 = \{0, 0, 0, 1, 2, 3, 3, 3\}$, respectively

function $\mathbf{F}$ is then defined, describing an invertible mapping to connect the parameter domain $\Omega_0 = (0, 1)^d$ with the physical domain Ω:

$$\mathbf{F} : \Omega_0 \to \Omega, \quad \mathbf{F}(\boldsymbol{\xi}) = (\mathbf{x}). \tag{8}$$

Provided that the physical domain Ω is topologically equivalent to the unit square, the geometry can be described by a single geometry function $\mathbf{F}$. In case of more complex geometries, a family of functions $\mathbf{F}^{(m)}$ $(m = 1, \ldots, K)$ is defined and we refer to Ω as a multipatch geometry consisting of K patches. For a more detailed description of the spatial discretization in IgA and multipatch constructions, the authors refer to chapter 2 of [1].

At each time step, we express u in Eq. (7) by a linear combination of multivariate B-spline basis functions of order p. Multivariate B-spline basis functions are defined as the tensor product of univariate B-spline basis functions $\phi_{i,p}$ $(i = 1, \ldots, N)$ which are uniquely defined on the parameter domain (0, 1) by an underlying knot vector $\Xi = \{\xi_1, \xi_2, \ldots, \xi_{N+p}, \xi_{N+p+1}\}$. Here, N denotes the number of B-spline basis functions and p the spline degree. Based on this knot vector, the basis functions are defined recursively by the Cox-de Boor formula [14], starting from the constant ones:

$$\phi_{i,0}(\xi) = \begin{cases} 1 & \text{if } \xi_i \le \xi < \xi_{i+1}, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

Higher-order B-spline basis functions of order $p > 0$ are then defined recursively:

$$\begin{aligned} \phi_{i,p}(\xi) = &\frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} \phi_{i,p-1}(\xi) \\ &+ \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} \phi_{i+1,p-1}(\xi). \end{aligned} \tag{10}$$

The resulting B-spline basis functions $\phi_{i,p}$ are non-zero on the interval $[\xi_i, \xi_{i+p+1})$ and possess the partition of unity property. Furthermore, the basis functions are C^{p-m_i} -continuous, where m_i denotes the multiplicity of knot ξ_i. Throughout this paper, we consider a uniform knot vector with knot span size h, where the first and last knot are repeated $p + 1$ times. As a consequence, the resulting B-spline basis functions are C^{p-1} continuous and interpolatory at both end points. Figure 1 illustrates both linear (left) and quadratic (right) B-spline basis functions based on such a knot vector.

As mentioned previously, the tensor product of univariate B-spline basis functions is adopted for the multidimensional case. Denoting the total number of multivariate B-spline basis functions $\Phi_{i,p}$ by N_{dof}, the solution u is thus approximated at each time step as follows:

$$u(\mathbf{x}) \approx u_{h,p}(\mathbf{x}) = \sum_{i=1}^{N_{\text{dof}}} u_i(t) \Phi_{i,p}(\mathbf{x}), \quad u_{h,p} \in \mathcal{V}_{h,p}. \tag{11}$$

Here, the spline space $\mathcal{V}_{h,p}$ is defined, using the inverse of the geometry mapping $\mathbf{F}^{-1}$ as pull-back operator, as follows:

$$\mathcal{V}_{h,p} := \text{span}\left\{\Phi_{i,p} \circ \mathbf{F}^{-1}\right\}_{i=1,\ldots,N_{\text{dof}}}. \tag{12}$$

By setting $v = \Phi_{j,p}$, Eq. (7) can be written as follows:

$$\begin{aligned} &(\mathbf{M} + \Delta t \theta \mathbf{K}) \mathbf{u}^{k+1} \\ &\quad = (\mathbf{M} - \Delta t (1 - \theta) \mathbf{K}) \mathbf{u}^k + \Delta t \mathbf{f}, \; k = 0, \ldots, N_t, \end{aligned} \tag{13}$$

where $\mathbf{M}$ and $\mathbf{K}$ denote the mass and stiffness matrix, respectively:

$$\mathbf{M}_{i,j} = \int_\Omega \Phi_{i,p} \Phi_{j,p} \, \mathrm{d}\Omega, \qquad \mathbf{K}_{i,j} = \int_\Omega \nabla \Phi_{i,p} \cdot \nabla \Phi_{j,p} \, \mathrm{d}\Omega. \tag{14}$$

3 Multigrid Reduction in Time

A traditional (i.e., sequential) time integration scheme would solve Eq. (13) for $k = 0, \ldots, N_t$ to obtain the numerical solution at each time instance. In this paper, however, we apply the Multigrid Reduction in Time (MGRIT) method to solve Eq. (13) parallel-in-time. For the ease of

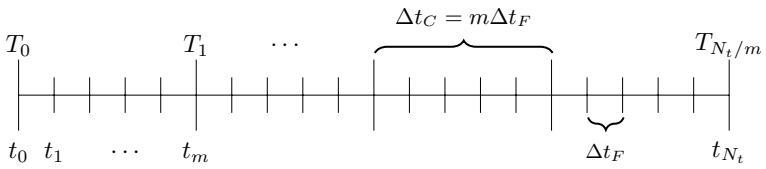

Fig. 2 Coarse and fine temporal mesh from 0 to T, based on Fig 2.1 of [6]

notation, we set $\theta = 1$ throughout the remainder of this section. Let $\Psi = (\mathbf{M} + \Delta t\mathbf{K})^{-1}$ denote the inverse of the left hand side operator. Then, Eq. (13) can be written as follows:

$$\mathbf{u}^{k+1} = \Psi\left(\mathbf{M}\mathbf{u}^k + \Delta t\mathbf{f}\right),\ k = 0, \dots, N_t, \tag{15}$$

$$= \Psi\mathbf{M}\mathbf{u}^k + \mathbf{g}^{k+1}, \quad k = 0, \dots, N_t, \tag{16}$$

where $\mathbf{g}^{k+1} = \Psi\Delta t\mathbf{f}$. Setting $\mathbf{g}^0$ equal to the initial condition $u^0(\mathbf{x})$ projected on the spline space $\mathcal{V}_{h,p}$, the time integration method can be written as a linear system of equations:

$$\mathbf{A}\mathbf{u} = \begin{bmatrix} I & & & \\ -\Psi\mathbf{M} & I & & \\ & \ddots & \ddots & \\ & & -\Psi\mathbf{M} & I \end{bmatrix} \begin{bmatrix} \mathbf{u}^0 \\ \mathbf{u}^1 \\ \vdots \\ \mathbf{u}^{N_t} \end{bmatrix} = \begin{bmatrix} \mathbf{g}^0 \\ \mathbf{g}^1 \\ \vdots \\ \mathbf{g}^{N_t} \end{bmatrix} = \mathbf{g}. \tag{17}$$

A sequential time integration scheme would correspond to a block-forward solve of this linear system of equations. In this paper, however, we adopt MGRIT to iteratively solve Eq. (17). First, we introduce the two-level MGRIT method, showing similarities with the well-known parareal algorithm [15]. In fact, it can be shown that both methods are equivalent, assuming a specific choice of relaxation [16]. Then, the multilevel variant of MGRIT will be presented in more detail.

3.1 Two-level MGRIT method

The two-level MGRIT method combines the use of a cheap coarse-level time integration method with an accurate more expensive fine-level one which can be performed in parallel. That is, the linear system of equations given by Eq. (17) can be solved iteratively by introducing a coarse temporal mesh with time step size $\Delta t_C = m\Delta t_F$. Here, Δt_F coincides with the Δt from the previous sections and m denotes the coarsening factor. Figure 2 illustrates both the fine and coarse temporal discretization.

The time instances $T_0, T_1, \dots, T_{N_t/m}$ are referred to as coarse points (or C-points), while the remaining points are called fine points (or F-points). The description of MGRIT is based on this division of the time instances in both coarse and fine points. By applying a numbering strategy that first numbers the F-points and then the C-points, we can write Eq. (17) as follows:

$$\begin{bmatrix} \mathbf{A}_{FF} & \mathbf{A}_{FC} \\ \mathbf{A}_{CF} & \mathbf{A}_{CC} \end{bmatrix} \begin{bmatrix} \mathbf{u}_F \\ \mathbf{u}_C \end{bmatrix} = \begin{bmatrix} \mathbf{g}_F \\ \mathbf{g}_C \end{bmatrix}, \tag{18}$$

where the matrix $\mathbf{A}$ can be decomposed as follows:

$$\begin{bmatrix} \mathbf{A}_{FF} & \mathbf{A}_{FC} \\ \mathbf{A}_{CF} & \mathbf{A}_{CC} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_F & \mathbf{0} \\ \mathbf{A}_{CF}\mathbf{A}_{FF}^{-1} & \mathbf{I}_C \end{bmatrix} \begin{bmatrix} \mathbf{A}_{FF} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{I}_F & \mathbf{A}_{FF}^{-1}\mathbf{A}_{FC} \\ \mathbf{0} & \mathbf{I}_C \end{bmatrix}, \tag{19}$$

where $\mathbf{I}_C$ and $\mathbf{I}_F$ are identity matrices. The 'ideal' restriction and prolongation operator are then defined as follows:

$$R = \begin{bmatrix} \mathbf{A}_{CF}\mathbf{A}_{FF}^{-1} & \mathbf{I}_C \end{bmatrix}, \qquad P = \begin{bmatrix} -\mathbf{A}_{FF}^{-1}\mathbf{A}_{FC} \\ \mathbf{I}_C \end{bmatrix}. \tag{20}$$

Within MGRIT, the 'ideal' prolongation operator P is typically adopted, while the 'ideal' restriction operator is replaced by $\tilde{R} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_C \end{bmatrix}$. The matrix $\mathbf{A}_{FF}$ is given by:

$$\mathbf{A}_{FF} = \begin{bmatrix} \mathbf{A}_\Psi & & \\ & \ddots & \\ & & \mathbf{A}_\Psi \end{bmatrix}, \quad \mathbf{A}_\Psi = \begin{bmatrix} I & & & \\ -\Psi\mathbf{M} & I & & \\ & \ddots & \ddots & \\ & & -\Psi\mathbf{M} & I \end{bmatrix}. \tag{21}$$

Note that each solve with $\mathbf{A}_\Psi$ corresponds to a single time step within a coarse interval, which is a completely independent process for each coarse interval and can therefore be performed in parallel. The Schur complement matrix $\mathbf{S}$ in Eq. (19) is given by:

$$\mathbf{S} = \mathbf{A}_{CC} - \mathbf{A}_{CF}\mathbf{A}_{FF}^{-1}\mathbf{A}_{FC} = \begin{bmatrix} I & & & \\ -(\Psi\mathbf{M})^m & I & & \\ & \ddots & \ddots & \\ & & -(\Psi\mathbf{M})^m & I \end{bmatrix}. \tag{22}$$

Instead of solving for $\mathbf{S}$ directly, MGRIT solves for a modified matrix $\tilde{\mathbf{S}}$ by replacing the operator $(\Psi\mathbf{M})^m \approx \Phi\mathbf{M}$, which corresponds to applying a single time step of a coarse time integrator. As a true multigrid method, the building blocks of the MGRIT method consist of *relaxation* (= fine time stepping) and a *coarse grid correction* (= coarse time stepping). Relaxation involves solving a linear system of the form

$$\mathbf{A}_{FF}\mathbf{u}_F = \mathbf{g}_F - \mathbf{A}_{FC}\mathbf{u}_C, \tag{23}$$

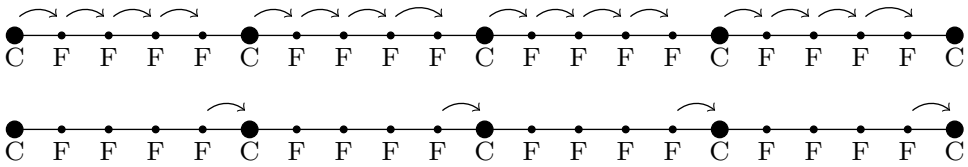

Fig. 3 Illustration of F-relaxation (top) and C-relaxation (bottom)

where $\mathbf{u}_F$ and $\mathbf{u}_C$ denote the solution at all *F*-points and *C*-points, respectively. Within relaxation, the solution is updated at the *F*-points based on the given values at the *C*-points. This time stepping from a coarse point *C* to all neighbouring fine points is also referred to as F-relaxation [6]. On the other hand, time stepping to a *C*-point from the previous F-point is referred to as *C*-relaxation. It should be noted that both types of relaxation are highly parallel and can be combined leading to so-called *CF*- or *FCF*-relaxation. Figure 3 illustrates both *C*- and F-relaxation methods.

The coarse grid correction involves solving the linear system of equations

$$\tilde{\mathbf{S}}\mathbf{u}_C = \tilde{R}(\mathbf{g} - \mathbf{A}\mathbf{u}), \tag{24}$$

which is a sequential procedure by design, but is much cheaper compared to the fine time integration (which can be performed in parallel). Here, the vector $\mathbf{u}_C$ is obtained by applying $\tilde{R}$ on $\mathbf{u}$.

3.2 Multilevel MGRIT method

The solution procedure described above can be extented to a true multilevel MGRIT method. First, we define a hierarchy of *L* temporal meshes, where the time step size for the discretization at level *l* ($l = 0, 1, \dots L$) is given by $\Delta t_F m^l$. The total number of levels *L* is related to the coarsening factor *m* and the total number of fine steps Δt_F by $L = \log_m(N_t)$. Let $\mathbf{A}^{(l)}\mathbf{u}^{(l)} = \mathbf{g}^{(l)}$ denote the linear system of equations based on the considered time step size at level *l*. The MGRIT method can then be written as follows:

1. Apply F-relaxation (= fine time stepping) on $\mathbf{A}^{(l)}\mathbf{u}^{(l)} = \mathbf{g}^{(l)}$:

   $$\mathbf{A}^{(l)}_{FF}\mathbf{u}^{(l)}_F = \mathbf{g}^{(l)}_F - \mathbf{A}^{(l)}_{FC}\mathbf{u}^{(l)}_C.$$

2. Determine the residual at level *l* and restrict it to level $l+1$ using the restriction operator $\tilde{R}$:

   $$\mathbf{r}^{(l+1)} = \tilde{R}\left(\mathbf{g}^{(l)} - \mathbf{A}^{(l)}\mathbf{u}^{(l)}\right).$$

3. Solve Eq. (24) (= coarse time stepping) to obtain $\mathbf{u}^{(l+1)}$:

   $$\tilde{\mathbf{S}}\mathbf{u}^{(l+1)} = \mathbf{r}^{(l+1)}.$$

4. Prolongate the correction using the 'ideal' interpolation operator *P* and update the solution at level *l*:

   $$\mathbf{u}^{(l)} := \mathbf{u}^{(l)} + P\mathbf{u}^{(l+1)}.$$

Recursive application of this scheme until the coarsest level is reached, leads to a so-called V-cycle. However, as with standard multigrid methods, alternative cycle types (i.e., W-cycles, F-cycles) can be defined. At all levels of the multigrid hierarchy, the operators are obtained by rediscretizing Eq. (1) using a different time step size.

4 *p*-multigrid method

Within the MGRIT algorithm, fine time stepping is performed in parallel within each time interval. Assuming a backward Euler time integration scheme, the following linear system of equations is solved within each time interval at every iteration:

$$(\mathbf{M} + \Delta t\mathbf{K})\mathbf{u}^{k+1} = \mathbf{M}\mathbf{u}^k + \Delta t\mathbf{f},\ k = 0, \dots, N_t. \tag{25}$$

Throughout this section we will omit the time step index *k* and write the linear system of equations given by Eq. (25) as follows:

$$\mathbf{A}_{h,p}\mathbf{u}_{h,p} = \mathbf{f}_{h,p}. \tag{26}$$

In a recent paper by the authors [12], this linear system of equations was solved within MGRIT by means of a (diagonally preconditioned) Conjugate-Gradient method. However, as the condition number of the system matrix increases exponentially in IgA with the spline degree *p*, the use of standard iterative solvers becomes less efficient for higher values of *p*. As a consequence, alternative solution techniques have been developed in recent years to overcome this dependency [17].

In this paper, we adopt a *p*-multigrid method [13] specifically designed for discretizations arising in IgA to solve the linear systems within MGRIT. Within the *p*-multigrid method, a low-order correction is obtained (at level $p = 1$) to update the solution at the high-order level. Starting from the high-order problem, the following steps are performed [13]:

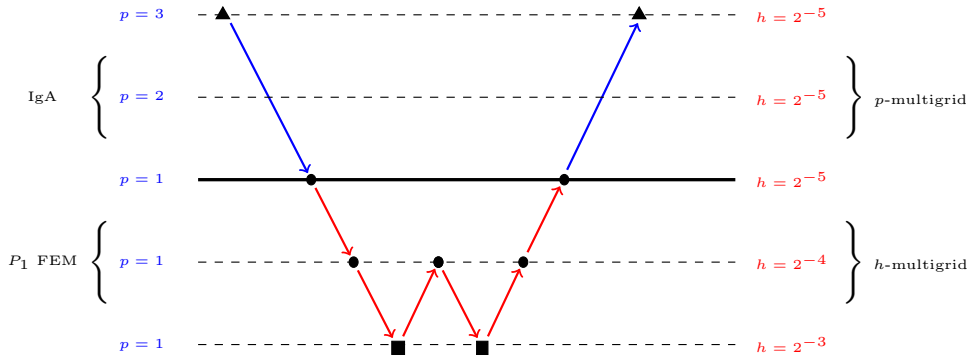

Fig. 4 Illustration of the *p*-multigrid method [13]. At $p = 1$, Gauss–Seidel is adopted as a smoother (filled circle), whereas at the high-order level ILUT is applied (filled triangle). At the coarsest level, a direct solver is applied to solve the residual equation (filled square)

1. Apply one presmoothing step to the initial guess $\mathbf{u}_{h,p}^0$:

$$\mathbf{u}_{h,p}^0 = \mathbf{u}_{h,p}^0 + \mathcal{S}_{h,p}\left(\mathbf{f}_{h,p} - \mathbf{A}_{h,p}\mathbf{u}_{h,p}^0\right), \tag{27}$$

   where $\mathcal{S}_{h,p}$ is a smoothing operator applied to the high-order problem.
2. Determine the residual at level p and project it onto the space $\mathcal{V}_{h,1}$ using the restriction operator $\mathcal{I}_p^1$:

$$\mathbf{r}_{h,1} = \mathcal{I}_p^1\left(\mathbf{f}_{h,p} - \mathbf{A}_{h,p}\mathbf{u}_{h,p}^0\right). \tag{28}$$

3. Solve the residual equation to determine the coarse grid error:

$$\mathbf{A}_{h,1}\mathbf{e}_{h,1} = \mathbf{r}_{h,1}. \tag{29}$$

4. Project the error $\mathbf{e}_{h,1}$ onto the space $\mathcal{V}_{h,p}$ using the prolongation operator $\mathcal{I}_1^p$ and update $\mathbf{u}_{h,p}^0$:

$$\mathbf{u}_{h,p}^0 := \mathbf{u}_{h,p}^0 + \mathcal{I}_1^p\left(\mathbf{e}_{h,1}\right). \tag{30}$$

5. Apply one postsmoothing step of the form (27) on the updated solution to obtain $\mathbf{u}_{h,p}^1$.

To approximately solve the residual equation given by Eq. (29) a single W-cycle of a standard *h*-multigrid method [18], using canonical prolongation and weighted restriction, is applied. As the level $p = 1$ corresponds to a low-order Lagrange discretization, an *h*-multigrid method (using Gauss–Seidel as a smoother) is known to be both efficient and cheap [19]. The resulting *p*-multigrid adopted throughout this paper is shown in Fig. 4.

Note that, we directly restrict the residual at the high-order level to level $p = 1$. This aggressive *p*-coarsening strategy has shown to significantly improve the computational efficiency of the resulting *p*-multigrid method [20], while maintaining its excellent convergence behavior.

Prolongation and restriction operators based on an L_2 projection are adopted to transfer vectors from the high-order level to the low-order level (and vice versa). These transfer operators have been used extensively in the literature [21–23] and are given by:

$$\mathcal{I}_1^p(\mathbf{v}_1) = (\mathbf{M}_p)^{-1}\mathbf{P}_1^p\,\mathbf{v}_1, \qquad \mathcal{I}_p^1(\mathbf{v}_p) = (\mathbf{M}_1)^{-1}\mathbf{P}_p^1\,\mathbf{v}_p. \tag{31}$$

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

$$\begin{aligned}(\mathbf{M}_p)_{(i,j)} &:= \int_\Omega \Phi_{i,p}\Phi_{j,p}\,\mathrm{d}\Omega,\\ (\mathbf{P}_1^p)_{(i,j)} &:= \int_\Omega \Phi_{i,p}\Phi_{j,1}\,\mathrm{d}\Omega,\end{aligned} \tag{32}$$

To prevent the explicit solution of a linear system of equations for each projection step, the consistent mass matrix in both transfer operators is replaced by its lumped counterpart by applying row-sum lumping. Note that, row-sum lumping can be applied within the variational formulation, due to the partition of unity and non-negativity of the B-spline basis functions.

Various choices can be made with respect to the smoother at the high-order level. The use of Gauss–Seidel or (damped) Jacobi as a smoother at level *p* leads to convergence rates of the resulting multigrid method that depend significantly on the spline degree *p* [24]. Alternative smoothers have been developed in recent years to overcome this shortcoming [25]. In particular, the use of ILUT factorizations [26] (i.e., as a preconditioner within a preconditioned Richardson iteration) has shown to be very effective in the context of IgA [24] and will therefore be adopted throughout the remainder of this paper. An efficient implementation of ILUT is available in the Eigen library [27]. Once the factorization $\mathbf{A}_{h,p} \approx \mathbf{L}_{h,p}\mathbf{U}_{h,p}$ is obtained, a single smoothing step is applied as follows:

$$\mathbf{e}_{h,p}^{(n)} = (\mathbf{L}_{h,p}\mathbf{U}_{h,p})^{-1}(\mathbf{f}_{h,p} - \mathbf{A}_{h,p}\mathbf{u}_{h,p}^{(n)}), \tag{33}$$

$$= \mathbf{U}_{h,p}^{-1}\mathbf{L}_{h,p}^{-1}(\mathbf{f}_{h,p} - \mathbf{A}_{h,p}\mathbf{u}_{h,p}^{(n)}), \tag{34}$$

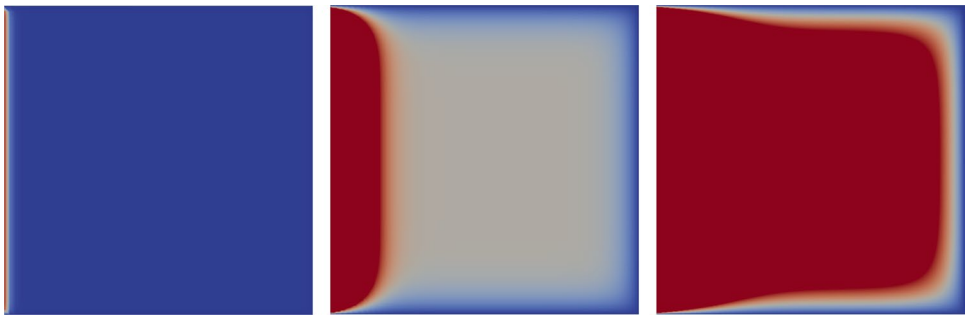

Fig. 5 Solution to the model problem at different times T using an inhomogeneous Neumann boundary condition at the left boundary using quadratic B-spline basis functions

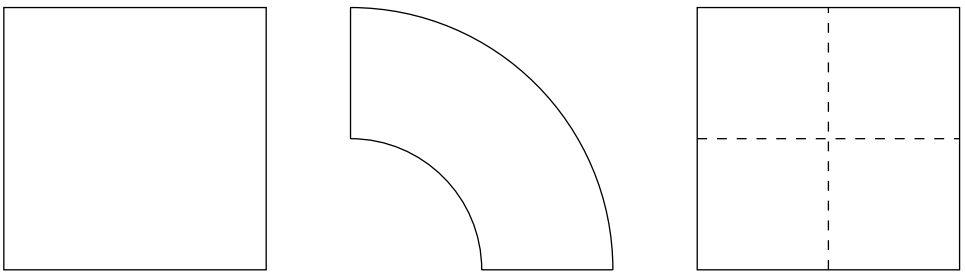

Fig. 6 Spatial domains Ω considered in [12]

$$\mathbf{u}_{h,p}^{(n+1)} = \mathbf{u}_{h,p}^{(n)} + \mathbf{e}_{h,p}^{(n)}, \tag{35}$$

The ILUT factorization is determined completely by a dropping tolerance τ and fill factor f. Based on previous studies by the authors, we choose $\tau = 10^{-12}$ and $f = 1$, which implies we only drop a few (very) small values during the factorization and $\mathbf{L}_{h,p}\mathbf{U}_{h,p}$ has a similar number of nonzero elements as $\mathbf{A}_{h,p}$.

5 Numerical results

To assess the quality of MGRIT when applied in combination with a p-multigrid method within Isogeometric Analysis, we consider the time-dependent heat equation in two dimensions given by Eq. (1). Figure 5 shows the resulting solution u at different time instances for $\Omega = [0, 1]^2$. Here, an inhomogeneous Neumann boundary condition is applied at the left boundary. Furthermore, the right-hand side is chosen equal to one and the initial condition is equal to zero.

Based on a spatial discretization with B-spline basis functions of order p and a mesh width h, MGRIT is applied to iteratively solve the resulting equation. Both the number of iterations and CPU timings needed to reach convergence will be investigated using both a (diagonally preconditioned) Conjugate Gradient method and the described p-multigrid method. Furthermore, we will investigate the parallel performance of MGRIT on modern computer architectures. The open-source C++ library G+Smo [28] is used to discretize the model problem in space using IgA, while, for the MGRIT algorithm, the parallel-in-time code XBraid, developed at Lawrence Livermore National Lab, is adopted [29]. The MGRIT method is said to have reached convergence if the relative residual (in the L_2 norm) at the end of an iteration is smaller or equal to 10^{-10}, unless stated otherwise.

As a starting point, we briefly summarize the results obtained in a previous paper of the authors (see [12]). There, numerical results were obtained for the same model problem using different hierarchies (i.e., a V-cycle, F-cycle and two-level method), time integration schemes (i.e., backward Euler, forward Euler and Crank–Nicolson) and domains of interest (see Fig. 6).

In general, it was observed that MGRIT converged in a low number (i.e., 5–10) of iterations, although the number of iterations was slightly higher when V-cycles were adopted instead of F-cycles or a two-level method. Furthermore, the number of iterations was independent of the mesh width h, spline degree of the B-spline basis functions p and the number of time steps N_t for all considered hierarchies and domains of interest. As expected from sequential time stepping methods, the use of the implicit backward Euler within MGRIT lead to the most stable time integration method. Finally, CPU timings were obtained for a limited number of processors, showing a strong dependency on the spline degree p when the Conjugate Gradient method was applied as a spatial solver within MGRIT.

In this section, we investigate the effect of using a p-multigrid method for the spatial solves compared to the use of a Conjugate Gradient method. Furthermore, we present numerical results when considering a three dimensional geometry (i.e., the unit cube). Finally, we investigate

Table 1 Number of MGRIT iterations for solving Eq. (1) on the unit square and a quarter annulus when adopting V-cycles for a varying number of time steps

	Unit square				Quarter annulus			
	$p=2$	$p=3$	$p=4$	$p=5$	$p=2$	$p=3$	$p=4$	$p=5$
$N_t = 250$	10	10	10	10	10	10	10	10
$N_t = 500$	10	10	10	10	10	10	10	10
$N_t = 1000$	11	11	11	11	11	11	11	11
$N_t = 2000$	11	11	11	11	11	11	11	11
$N_t = 250$	10	10	10	10	10	10	10	10
$N_t = 500$	10	10	10	10	10	10	10	10
$N_t = 1000$	11	11	11	11	11	11	11	11
$N_t = 2000$	11	11	11	11	11	11	11	11

Here, *p*-multigrid (top) and the CG method (bottom) are adopted for the spatial solves and backward Euler for the time integration

Table 2 Number of MGRIT iterations for solving Eq. (1) on the unit square and a quarter annulus when adopting V-cycles for varying mesh widths

	Unit square				Quarter annulus			
	$p=2$	$p=3$	$p=4$	$p=5$	$p=2$	$p=3$	$p=4$	$p=5$
$h = 2^{-6}$	9	9	9	9	9	9	9	9
$h = 2^{-7}$	9	9	9	9	9	9	9	9
$h = 2^{-8}$	10	10	10	10	9	9	9	9
$h = 2^{-9}$	10	10	10	10	10	10	10	10
$h = 2^{-6}$	9	9	9	9	9	9	9	9
$h = 2^{-7}$	9	9	9	9	9	9	9	9
$h = 2^{-8}$	10	10	10	10	9	9	9	9
$h = 2^{-9}$	10	10	10	10	10	10	10	10

Here, *p*-multigrid (top) and the CG method (bottom) are adopted for the spatial solves and backward Euler for the time integration

Table 3 Number of MGRIT iterations for solving Eq. (1) on the unit cube when adopting V-cycles for a varying number of time steps

	$p=2$	$p=3$	$p=4$	$p=5$		$p=2$	$p=3$	$p=4$	$p=5$
$N_t = 250$	10	10	10	11	$h = 2^{-3}$	9	9	9	10
$N_t = 500$	11	11	11	11	$h = 2^{-4}$	9	9	9	10
$N_t = 1000$	11	11	11	11	$h = 2^{-5}$	10	10	10	10
$N_t = 2000$	11	11	11	11	$h = 2^{-6}$	10	10	10	10

the weak and strong scaling of MGRIT on modern architectures when applied in the context of IgA. As this research focuses on the spatial solver and scalability, we will restrict ourselves to the backward Euler method and the use of V-cycles within MGRIT.

5.1 Iteration numbers

As a first step, we compare the number of MGRIT iterations to reach convergence when a *p*-multigrid method or a (diagonally preconditioned) Conjugate Gradient method is adopted while keeping all other parameters the same. Table 1 shows the results when the mesh width is kept constant ($h = 2^{-6}$) for the unit square and a quarter annulus when adopting V-cycles with a *p*-multigrid (top) and CG method (bottom), respectively. For both benchmarks and all configurations, the number of iterations needed with MGRIT to reach convergence is independent of the number of time steps N_t and spline degree p. Furthermore, the number of MGRIT iterations is identical when adopting a *p*-multigrid method compared to the use of a Conjugate Gradient method.

Table 2 shows the results for different values of the mesh width h when the number of time steps is kept constant ($N_t = 100$) for both benchmarks when adopting V-cycles. The number of MGRIT iterations is independent of the mesh width h and spline degree p. Furthermore, the number of MGRIT iterations is identical when adopting a *p*-multigrid method compared to the use of a Conjugate Gradient method.

Table 4 Number of MGRIT iterations for solving Eq. (1) on the unit square using forward Euler ($\theta = 0$) and Crank–Nicolson ($\theta = 0.5$) when adopting *V*-cycles

	Forward Euler				Crank–Nicolson			
	$p = 2$	$p = 3$	$p = 4$	$p = 5$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
$h = 2^{-3}$	13	13	13	14	11	11	11	12
$h = 2^{-4}$	13	13	*	*	11	11	11	11
$h = 2^{-5}$	*	*	*	*	11	11	13	23
$h = 2^{-6}$	*	*	*	*	13	28	52	88

Results when adopting the *p*-multigrid method have been obtained for a three-dimensional benchmark problem as well. Table 3 shows the number of MGRIT iterations for different values of N_t, p and h when the unit cube is considered as geometry. In general, the number of iterations needed to reach convergence is independent of the number of time steps, spline degree and mesh width. Furthermore, the number of iterations are comparable to the ones obtained for the two-dimensional benchmark problems.

Finally, we investigate the influence of the time integration scheme on the number of MGRIT iterations. Table 4 shows the number of MGRIT iterations needed to reach convergence for the forward Euler ($\theta = 0$) and Crank–Nicolson ($\theta = 0.5$) method. Results can be compared to the ones obtained with the backward Euler method (see Table 2). For many configurations, MGRIT using forward Euler does not convergence (which is related to the CFL condition), while the Crank–Nicolson method converges for all configurations. A small dependency on h and p is, however, visible. Based on these results, the backward Euler method will be adopted throughout the remainder of this paper. For a more detailed analysis regarding different time integration schemes within MGRIT, the authors refer to [12].

Although the number of MGRIT iterations is identical for all configurations when adopting a *p*-multigrid or Conjugate Gradient method for solving the linear systems of equations, it is expected that CPU timings will differ significantly. Therefore, focus will lie on CPU timings throughout the remainder of this section.

5.2 CPU timings

CPU timings have been obtained when a *p*-multigrid method or Conjugate Gradient method is adopted for the spatial solves within MGRIT. As in the previous section, we adopt V-cycles, a mesh width of $h = 2^{-6}$ and the unit square as our domain of interest. Note that the corresponding iteration numbers can be found in Table 1. The computations are performed on three compute nodes each consisting of an Intel(R) i7-10700 (@ 2.90GHz) Cometlake processor with 8 hardware cores (hyperthreading turned on) and 128GB DDR4 main memory organized in 4 modules of 32GB each.

Figure 7 shows the CPU time needed to reach convergence for a varying number of cores, a different number of time steps and different values of *p*. When the Conjugate Gradient method is adopted for the spatial solves, doubling the number of time steps leads to an increase of the CPU time by a factor of two. Furthermore, it can be observed that the CPU timings significantly increase for higher values of *p* which is related to the spatial solves required at every time step. As standard iterative solvers (like the Conjugate Gradient method) have a detoriating performance for increasing values of *p*, more iterations are required to reach convergence for each spatial solve, resulting in higher computational costs of the MGRIT method. When focussing on the number of cores, it can be seen that doubling the number of cores significantly reduces the CPU time needed to reach convergence. More precisely, a reduction of 45–50% can be observed when doubling the number of cores to 6, implying the MGRIT algorithm is highly parallelizable.

As with the use of the Conjugate Gradient method, doubling the number of time steps leads to an increase of the CPU time by a factor of two when a *p*-multigrid method is adopted. For $p = 2$, the use of a *p*-multigrid method leads to higher CPU timings compared to the use of the Conjugate Gradient method for all values of N_t. However, the dependency of the CPU timings on the spline degree is significantly mitigated, which leads to a serious decrease of the CPU timings compared to the use of the Conjugate Gradient method when higher values of *p* are considered. For example, for $N_t = 2000$ and $p = 5$ a speed-up of more than a factor of 10 is achieved.

Again, increasing the number of cores from 3 to 6, reduces the CPU time needed to reach convergence by 45–50%. These results show that MGRIT combined with a *p*-multigrid method leads to an overall more efficient method. Therefore, a larger computer cluster will be considered in the next section to further investigate the scalability of MGRIT (i.e., weak and strong scalability) when combined with a *p*-multigrid method within IgA.

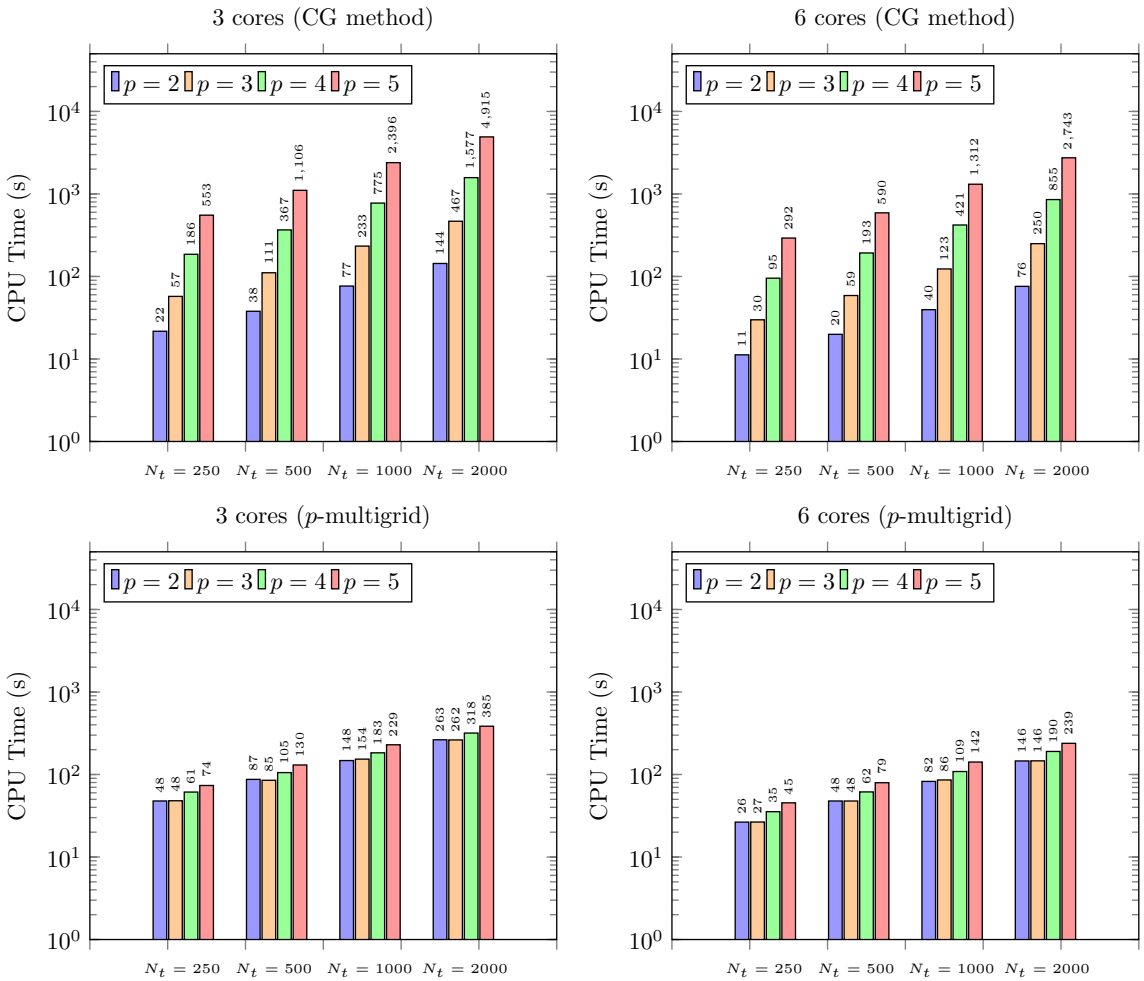

Fig. 7 CPU timings for MGRIT using V-cycles and backward Euler on the unit square for a fixed problem size ($h = 2^{-6}$) adopting a different number of processors. Here the Conjugate Gradient method (top) and a *p*-multigrid method (bottom) are used for all spatial solves within MGRIT

6 Scalability

In the previous sections, we applied MGRIT adopting a relatively low number of cores. Here, it was shown that the use of a *p*-multigrid method significantly reduces the dependency of the CPU timings on the spline degree. In this section, we investigate the scalability of MGRIT (combined with a *p*-multigrid method) on a modern architecture. More precisely, we will investigate both strong and weak scalability on the Lisa system, one of the nationally used clusters of the Netherlands[1].

6.1 Strong scalability

First, we fix the total problem size and increase the number of cores (i.e., strong scalability). That is, we consider the same benchmark problem as in the previous sections,

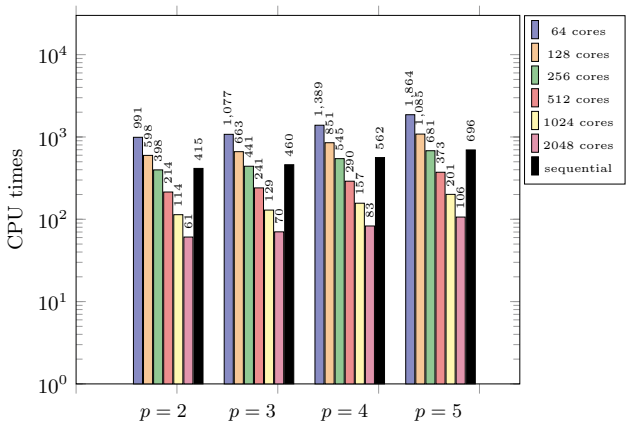

Fig. 8 Strong scalability study for MGRIT using V-cycles and backward Euler on the unit square. Here *p*-multigrid is used for all spatial solves within MGRIT

[1] https://userinfo.surfsara.nl/systems/lisa.

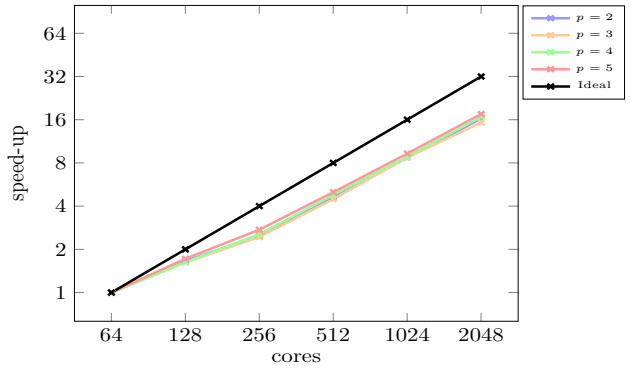

Fig. 9 Speed-up with MGRIT using V-cycles and backward Euler on the unit square. Here *p*-multigrid is used for all spatial solves within MGRIT

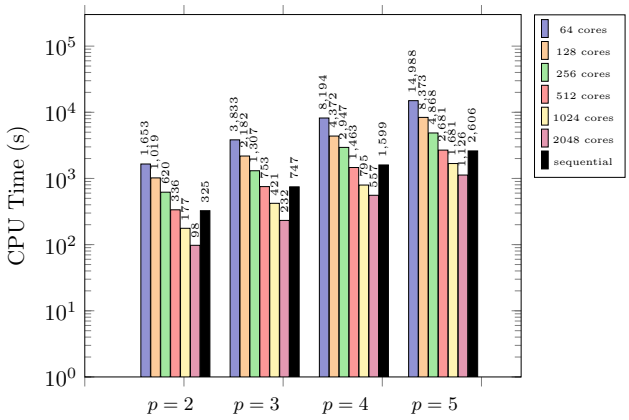

Fig. 10 Strong scalability study for MGRIT using V-cycles and backward Euler on the unit cube. Here *p*-multigrid is used for all spatial solves within MGRIT

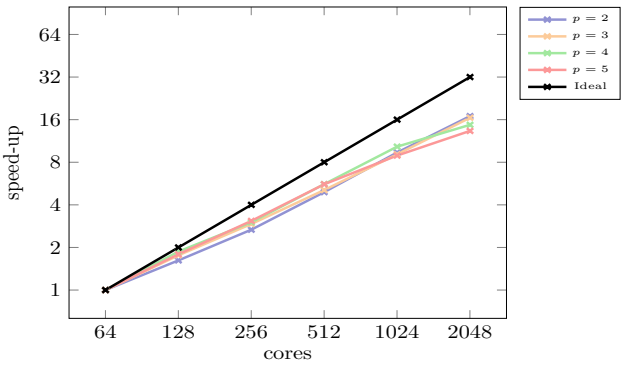

Fig. 11 Speed-up with MGRIT using V-cycles and backward Euler on the unit cube. Here *p*-multigrid is used for all spatial solves within MGRIT

but with a mesh width of $h = 2^{-6}$ and a number of time steps N_t of 10.000. As before, backward Euler is applied for the time integration and V-cycles are adopted as MGRIT hierarchy. Figure 8 shows the CPU timings needed to reach convergence for a varying number of Intel Xeon Gold 6130 (@ 2.10GHz) processors, where each processor consists of 16 cores. For all values of *p*, increasing the number of cores leads to significant speed-ups which illustrates the parallizability of the MGRIT method up to 2048 cores. To compare the results with a sequential time integration method, results with a backward Euler method have been added as well. Here, the CPU timings are independent of the number of processors and shown in the most right column for each value of *p* ('sequential'). Clearly, MGRIT outperforms the sequential algorithm when the number of cores is larger or equal to 128. This behavior has been observed in the literature as well in case of a finite difference discretization for a similar model problem, see [6].

Figure 9 shows the obtained speed-ups as a function of the number of cores for different values of *p* based on the results presented in Fig. 8. As a comparison, the ideal speed-up has been added, assuming a perfect parallizability of the MGRIT method. Note that, for all values of *p*, the observed speed-up slightly increases when the number of cores is higher than 256. Furthermore, the obtained speed-ups remain high, even when the number of cores is further increased to 2048, and is independent of the spline degree *p*.

Strong scalability has been investigated for the three-dimensional benchmark problem as well. Figure 10 shows the strong scalability for MGRIT on the unit cube. In general, the obtained results are comparable to the ones obtained in two dimensions, showing significant speed-ups when increasing the number of cores for all values of *p*. Results obtained with a sequential time integration method have been added as well, showing comparable results to MGRIT when adopting 512 cores. It should be noted that, compared to the two-dimensional benchmark problem, the CPU timings grow significantly faster for increasing values of *p*. This is well-known in Isogeometric Analysis [30] and is related to the relatively high number of nonzero entries in three dimensions when considering higher values of *p*. The use of the (preconditioned) Conjugate Gradient method would even lead to a significant higher growth in CPU timings, as the number of iterations needed to reach convergence for every spatial solve increases excessively in three dimensions when adopting a standard solver.

Figure 11 shows the obtained speed-ups for different values of *p* based on the results presented in Fig. 10. The obtained speed-ups are similar to the ones obtained for the two-dimensional problem but vary slightly more for different values of *p*. In general, the observed speed-ups remain high, even when the number of cores is increased to 2048.

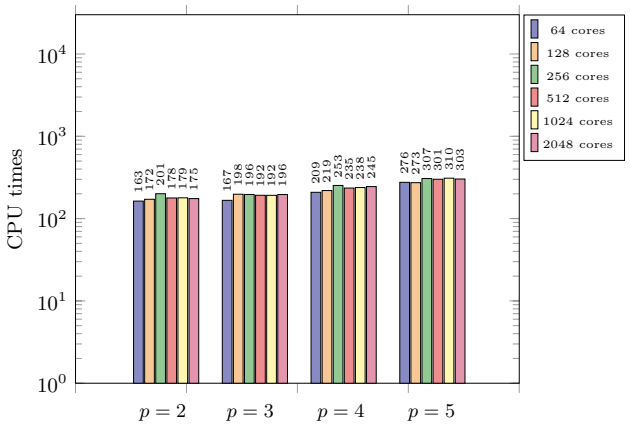

Fig. 12 Weak scalability study for MGRIT using V-cycles and backward Euler on the unit square. Here *p*-multigrid is used for all spatial solves within MGRIT

6.2 Weak scalability

As a next step, we consider the unit square as our domain of interest but keep the problem size per processor fixed (i.e. weak scalability). In case of 64 cores, the number of time steps equals 1000 and is adjusted based on the number of cores. Figure 12 shows the CPU time needed to reach convergence for a different number of cores and different values of *p*. Clearly, the CPU timings remain (more or less) constant when the number of cores is increased, showing the weak scalability of the MGRIT method. Although the CPU timings slightly increase for higher values of *p*, the strong *p*-dependency observed with the Conjugate Gradient method is clearly mitigated.

7 Conclusions

In this paper, we combined MGRIT with a *p*-multigrid method for discretizations arising in Isogeometric Analysis. Numerical results obtained for a variety of benchmark problems show that the use of a *p*-multigrid method for all spatial solves within MGRIT results in convergence rates independent of the mesh width *h*, spline degree *p* and number of time steps N_t. Furthermore, CPU timings depend only mildly on the spline degree *p* in two dimensions. This is in sharp contrast to standard solvers (e.g. a Conjugate Gradient method), which show a deteriorating performance (in terms of CPU timings) for higher values of *p* already in two dimensions. Furthermore, the obtained CPU timings when adopting a *p*-multigrid method are significantly lower for almost all considered configurations. On modern computer architectures, both strong and weak scalability of the resulting MGRIT method have been investigated, showing good scalability up to 2048 cores, illustrating the potential of MGRIT (combined with a *p*-multigrid method) for time-dependent simulations in IgA.

Within this paper, we restrict ourselves to first- and second-order accurate time integration schemes. As the use of high-order B-spline basis functions significantly reduces the spatial discretization error, the use of alternative (and in particular higher-order) time integration scheme is interesting and will be investigated in future work. Furthermore, we will focus on the application of MGRIT to more challenging benchmark problems, in particular those where IgA has proven to be a viable alternative to FEM.

Author Contributions All authors contributed to the study conception and design. All authors read and approved the final manuscript.

Funding The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Hughes TJR, Cottrell JA, Bazilevs Y (2005) Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Comput Methods Appl Mech Eng 194:4135–4195. https://doi.org/10.1016/j.cma.2004.10.008
2. Cottrell J, Reali A, Bazilevs Y, Hughes T (2006) Isogeometric analysis of structural vibrations. Comput Methods Appl Mech Eng 195(41–43):5257–5296. https://doi.org/10.1016/j.cma.2005.09.027
3. Bazilevs Y, Calo VM, Zhang Y, Hughes TJR (2006) Isogeometric fluid-structure interaction analysis with applications to arterial blood flow. Comput Mech 38(4–5):310–322. https://doi.org/10.1007/s00466-006-0084-3
4. Wall WA, Frenzel MA, Cyron C (2008) Isogeometric structural shape optimization. Comput Methods Appl Mech Eng 197(33–40):2976–2988. https://doi.org/10.1016/j.cma.2008.01.025
5. Hughes TJR, Reali A, Sangalli G (2008) Duality and unified analysis of discrete approximations in structural dynamics and wave

propagation: Comparison of *p*-method finite elements with *k*-method NURBS. Comput Methods Appl Mech Eng 197:4104–4124. https://doi.org/10.1016/j.cma.2008.04.006
6. Falgout RD, Friedhoff S, Kolev TV, MacLachlan SP, Schroder JB (2014) Parallel time integration with multigrid. SIAM J Sci Comput 36(6):635–661. https://doi.org/10.1137/130944230
7. Ries M, Trottenberg U, Winter G (1983) A note on MGR methods. Linear Algebra Appl 49:1–26. https://doi.org/10.1016/0024-3795(83)90091-5
8. Langer U, Moore SE, Neumüller M (2016) Space-time isogeometric analysis of parabolic evolution problems. Comput Methods Appl Mech Eng 306:342–363. https://doi.org/10.1016/j.cma.2016.03.042
9. Dobrev VA, Kolev T, Petersson NA, Schroder JB (2017) Two-level convergence theory for multigrid reduction in time (MGRIT). SIAM J Sci Comput 39(5):501–527. https://doi.org/10.1137/16m1074096
10. Günther S, Gauger NR, Schroder JB (2018) A non-intrusive parallel-in-time approach for simultaneous optimization with unsteady PDEs. Optim Methods Softw 34(6):1306–1321. https://doi.org/10.1080/10556788.2018.1504050
11. Lecouvez M, Falgout RD, Woodward CS, Top P (2016) A parallel multigrid reduction in time method for power systems. In: 2016 IEEE power and energy society general meeting (PESGM), pp 1–5. https://doi.org/10.1109/PESGM.2016.7741520
12. Tielen R, Möller M, Vuik C (2021) Multigrid reduced in time for isogeometric analysis. In: VI Eccomas Young Investigators Conference
13. Tielen R, Möller M, Göddeke D, Vuik C (2020) *p*-multigrid methods and their comparison to *h*-multigrid methods within isogeometric analysis. Comput Methods Appl Mech Eng. https://doi.org/10.1016/j.cma.2020.113347
14. De Boor C (1978) A practical guide to splines. Springer, New York
15. Lions J-L (2001) Résolution d'edp par un schéma en temps pararéel a parareal in time discretization of pde's. CRASM 332(7):661–668. https://doi.org/10.1016/S0764-4442(00)01793-6
16. Gander MJ, Vandewalle S (2007) Analysis of the parareal time-parallel time-integration method. SIAM J Sci Comput 29(2):556–578. https://doi.org/10.1137/05064607x
17. Donatelli M, Garoni C, Manni C, Capizzano S, Speleers H (2017) Symbol-based multigrid methods for galerkin B-spline isogeometric analysis. SIAM J Numer Anal 55:31–62. https://doi.org/10.1137/140988590
18. Hackbush W (1985) Multi-grid methods and applications. Springer, Berlin. https://doi.org/10.1007/978-3-662-02427-0
19. Trottenberg U, Oosterlee C, Schüller A (2001) Multigrid. Academic Press, London
20. Tielen R, Möller M, Vuik K (2021) A direct projection to low-order level for *p*-multigrid methods in isogeometric analysis. In: Vermolen F, Vuik C (eds) Numerical mathematics and advanced applications, ENUMATH 2019 - European Conference. Lecture notes in computational science and engineering, pp 1001–1009. Springer, Cham. https://doi.org/10.1007/978-3-030-55874-1_99
21. Briggs WL, Henson VE, McCormick SF (2000) A multigrid tutorial, 2nd edn. SIAM, Philadelphia. https://doi.org/10.1137/1.9780898719505
22. Brenner SC, Scott LR (1994) The mathematical theory of finite element methods. Springer, New York
23. Sampath RS, Biros G (2010) A parallel geometric multigrid method for finite elements on octree meshes. SIAM J Sci Comput 32:1361–1392. https://doi.org/10.1137/090747774
24. Tielen R, Möller M, Vuik C (2018) Efficient multigrid based solvers for isogeometric analysis. In: van Brummelen H, Vuik C, Möller M, Verhoorsel C, Simeon B, Jüttler B (eds.), Isogeometric analysis and applications 2018. Lecture notes in computational science and engineering. Springer, Cham. https://doi.org/10.1007/978-3-030-49836-8
25. Hofreither C, Takacs S, Zulehner W (2017) A robust multigrid method for isogeometric analysis in two dimensions using boundary correction. Comput Methods Appl Mech Eng 316:22–42. https://doi.org/10.1016/j.cma.2016.04.003
26. Saad Y (1994) ILUT: a dual threshold incomplete LU factorization. Numer Linear Algebra Appl 1:387–402. https://doi.org/10.1002/nla.1680010405
27. Guennebaud G et al (2010) Eigen v3. http://eigen.tuxfamily.org
28. Mantzaflaris A et al (2018) G+Smo (Geometry plus Simulation modules) v0.8.1. http://github.com/gismo
29. XBraid: Parallel multigrid in time. http://llnl.gov/casc/xbraid
30. Collier N, Dalcin L, Pardo D, Calo V (2013) The costs of continuity: performance of iterative solvers on isogeometric finite elements. SIAM J Sci Comput 35:767–784. https://doi.org/10.1137/120881038