ESAIM: M2AN 51 (2017) 1173–1195
DOI: 10.1051/m2an/2016055

ESAIM: Mathematical Modelling and Numerical Analysis
www.esaim-m2an.org

PRECONDITIONERS FOR THE DISCONTINUOUS GALERKIN TIME-STEPPING METHOD OF ARBITRARY ORDER

Steffen Basting[1] and Eberhard Bänsch[2]

Abstract. We develop a preconditioner for systems arising from space-time finite element discretizations of parabolic equations. The preconditioner is based on a transformation of the coupled system into block diagonal form and an efficient solution strategy for the arising 2×2 blocks. The suggested strategy makes use of an inexact factorization of the Schur complement of these blocks, for which uniform bounds on the condition number can be proven. The main computational effort of the preconditioner lies in solving implicit Euler-like problems, which allows for the usage of efficient standard solvers. Numerical experiments are performed to corroborate our theoretical findings.

Mathematics Subject Classification. 65M12, 65M60.

Received 22 September, 2015. Revised 17 June, 2016. Accepted 24 August, 2016.

1. Introduction

Using low order time discretization schemes like backward Euler or Crank−Nicolson is rather standard practice when solving time dependent partial differential equations (PDE). However, in most cases one can take great advantage of using higher order discretization schemes. Obvious choices for such discretization schemes are those frequently used for discretizing ordinary differential equations (ODE) like multistep or Runge−Kutta methods. These approaches have been studied for instance in [13, 34] and the references therein.

Another approach is to use *variational* time discretization schemes, more particularly to use continuous or discontinuous Galerkin schemes in time. Approximations of this type possess many appealing features, let us just mention three of them:

- Stability: many variational time discretization approaches exhibit favourable stability properties like A-stability. In particular, for the dG(k) method considered in this paper, even strong A-stability can be shown to hold for arbitrary polynomial degrees k [18, 27].
- Convergence properties: many variational time discretization schemes exhibit super convergence properties, for instance dG(k), which is super convergent of order $2k + 1$ at the nodal points [34].
- Generality: being of variational type, time discretization is treated similar to space discretization. As a consequence, many of the techniques developed for space discretization over the years are directly applicable to time discretization.

Keywords and phrases. Finite element method, time discretization, discontinuous Galerkin, preconditioning.

[1] Institute of Applied Mathematics (LS III), TU Dortmund, Vogelpothsweg 8, 44227 Dortmund, Germany.
Steffen.Basting@math.tu-dortmund.de

[2] Applied Mathematics III, Dept. of Mathematics, Cauerstr. 11, 91058 Erlangen, Germany

Article published by EDP Sciences

If combined with Galerkin schemes for spatial discretization like finite elements, they also offer a clean way for *a posteriori* error control, see for instance [2, 7, 20]. Naturally, coupled space-time systems occur in (time-dependent) optimal control problems, *e.g.* [1, 31, 32] to name a few. Moreover variational time discretization schemes are particularly well suited to discretize problems on time-dependent domains, see [4, 5].

However, Galerkin time discretization is not that popular among practitioners. The reason for the reluctance to use this type of discretization might be found in the considerably more complicated discrete systems arising after full discretization. Using a Galerkin time discretization with, say, $r \in \mathbb{N}$ degrees of freedom per time step results in a *coupled* system of r (spatially discretized) PDEs to be solved in each time step. At first glance it is not obvious how to solve or precondition these systems efficiently.

For the numerical solution of the coupled system arising from variational time discretization schemes, various techniques were proposed: Schötzau *et al.* [28, 29, 37] consider the dG(k) methods for time-independent linear operators and decouple the arising system into linear problems which have the same structure as as an implicit Euler-like discretization of the system, but are complex valued. In [26] Richter et al. consider a solution strategy for nonlinear parabolic problems discretized by the dG(k) method based on an inexact factorization of the block eliminated dG(k) system. This approach turns out to be quite costly for linear problems, however. For the dG(1) case, Hussain et al. consider an efficient solution approach based on a multigrid preconditioned BiCGStab solver in [15].

At this point, it seems important to emphasize that variational time discretization schemes share many similarities with implicit Runge−Kutta methods, see for instance [2] for a unified theoretical treatment of these methods. Therefore, preconditioners used for the solution of systems arising from implicit Runge−Kutta methods should be mentioned as well. Preconditioners based on the W-transformation [13] of the Runge−Kutta coefficient matrix were discussed in [16, 17]. Mardal et al. analyze a block diagonal preconditioner for A-stable Runge−Kutta schemes in [21] and show order-optimality, *i.e.* no dependency of the condition number on space- and time discretization parameters. However, the suggested preconditioner is not order-optimal with respect to the number of Runge−Kutta stages. In a preceding publication [30], the same authors showed numerically that this dependency can be weakened by a Gauss−Seidel type preconditioner. We would like to emphasize that in view of the aforementioned similarities between implicit Runge−Kutta and the variational time discretization methods considered in this paper, our results carry over to implicit Runge−Kutta methods as well.

In the present paper we propose a strategy to transform the arising systems such that, per time step, only *decoupled* real-valued problems of a single (spatially discretized) PDE or a block 2×2 system at most have to be solved. Hereby, the single system or one block of the system is equivalent to an implicit Euler discretization of the underlying parabolic problem. There is an abundance of efficient solvers like for instance multigrid to tackle such a problem.

Thus, assuming there is an optimal standard solver for the Euler-discretized problem at hand, it remains to effectively precondition the 2×2 systems. To this end, we propose a strategy based on the ideas in [3] for preconditioning fourth order (in space) problems. In order to avoid complex arithmetic (which may not be provided by the underlying finite element code to be used), the main ingredient of our strategy is an inexact factorization of the Schur complement of the 2×2 system. In this sense, the approach can be seen as a generalization of a similar Schur complement approach for the dG(1) method that was presented in [36] to arbitrary degree k. For the preconditioned operator, uniform bounds of the condition number with respect to space and time discretization parameters as well as the degree of the method can be proven.

Let us define the problem under consideration in more detail. Consider the following parabolic equation on a given time interval $I := (0, T)$ and a bounded domain $\Omega \subset \mathbb{R}^d, d \in \{2, 3\}$: *Find* $u : I \times \Omega \to \mathbb{R}$ *such that*

$$\left.\begin{aligned} \partial_t u(t, \mathbf{x}) - \nabla \cdot (\mathbf{D}(\mathbf{x}) \nabla u(t, \mathbf{x})) &= f(t, \mathbf{x}) && \forall (t, \mathbf{x}) \in I \times \Omega \\ u(t, \mathbf{x}) &= 0 && \forall (t, \mathbf{x}) \in I \times \partial\Omega \\ u(0) &= u_0 && \text{in } \Omega, \end{aligned}\right\} \tag{1.1}$$

where $\mathbf{D}_{i,j}, i, j = 1, \ldots, d$ are given coefficients and $f : I \times \Omega \to \mathbb{R}$, $u_0 : \Omega \to \mathbb{R}$ are given data.

We will use a weak formulation of equation (1.1) in both space and time. For that purpose, we consider the space of square integrable functions $L^2(\Omega)$, the Sobolev space of once weakly differentiable functions $\mathbb{V} := H_0^1(\Omega) = \{u \in H^1(\Omega) : \ u_{|\partial\Omega} = 0\}$, its dual space $\mathbb{V}^* = H^{-1}(\Omega)$.

Furthermore, we define the parabolic function space

$$X := \{v \in L^2((0,T),\mathbb{V}) : \ \partial_t v \in L^2((0,T),\mathbb{V}^*)\}. \tag{1.2}$$

The space X is continuously embedded into $C^0([0,T],L^2(\Omega))$ and thus, for functions $v \in X$, we also use the notation $v(t)$ to denote $v(t,\cdot)$ for $t \in [0,T]$.

Throughout this paper, we make the assumption that $\mathbf{D}_{i,j} \in L^\infty(\Omega), i,j = 1,\dots,d$, and that the corresponding coefficient matrix $\mathbf{D}$ is symmetric and uniformly positive definite. If the given data fulfills the regularity assumptions

$$u_0 \in L^2(\Omega), \ f \in L^2((0,T),L^2(\Omega)),$$

existence and uniqueness of a weak solution $u \in X$ to problem (1.1) can be shown (see *e.g.* Chap. 7.1, Thms. 3 and 4 in [9], Thm. 1.1 in [33]). In particular, due to the embedding $X \hookrightarrow C^0([0,T],L^2(\Omega))$, the initial condition $u(0) = u_0$ in L^2 is well defined.

Let us introduce the bilinear form $a : \mathbb{V} \times \mathbb{V} \to \mathbb{R}$ which is defined as

$$a(u,v) := \int_\Omega \mathbf{D}\nabla u \cdot \nabla v \,\mathrm{d}\boldsymbol{x}. \tag{1.3}$$

With these notations, a weak formulation to problem (1.1) can be stated as follows: *let* $f \in L^2((0,T),L^2(\Omega))$ *and* $u_0 \in L^2(\Omega)$ *be given. Find* $u \in X$ *such that*

$$\int_0^T \langle \partial_t u, v\rangle \,\mathrm{d}t + \int_0^T a(u,v)\,\mathrm{d}t + (u(0),v(0)) = \int_0^T (f,v)\,\mathrm{d}t + (u_0,v(0)) \tag{1.4}$$

for all $v \in X$.

Here, $(\cdot,\cdot)$ denotes the L^2-inner product and $\langle\cdot,\cdot\rangle$ is the duality pairing on $\mathbb{V}$.

The rest of this paper is organized as follows. Section 2 deals with the variational time discretization of equation (1.4) by a discontinuous Galerkin (dG) method and space discretization by conforming finite elements. For the resulting fully discrete system, an efficient solution strategy is developed in Section 3. We begin by transforming the coupled system into a system of block diagonal form consisting of single blocks of implicit Euler-like problems, and coupled 2×2 blocks whose efficient treatment is crucial for the overall efficiency of the method. In order to circumvent complex arithmetic, we make use of a preconditioned Schur complement formulation. The occurring ill-conditioned fourth order operator is preconditioned by means of an inexact factorization for which uniform bounds on the condition number can be derived. We touch on numerical realization of the preconditioner and our implementation in Section 4 and conclude with numerical experiments in Section 5.

2. Variational time discretization for parabolic partial differential equations

In order to discretize equation (1.4) in time, the time interval $I = (0,T)$ is subdivided into N subintervals $I_n = (t_{n-1},t_n]$, where $0 = t_0 < t_1 < \cdots < t_N = T$. The time step size is denoted by $\tau_n := t_n - t_{n-1}$.

The general idea of variational time discretization is to approximate the function $u : I \to \mathbb{V}$ by a piecewise polynomial function u_τ where $u_{\tau|I_n} \in \mathbb{P}_k(I_n,\mathbb{V})$. To this end, define the space

$$\mathbb{P}_k^{\mathrm{dc}} := \{v \in L^2((0,T),\mathbb{V}) : \ v_{|I_n} \in \mathbb{P}_k(I_n,\mathbb{V}) \ \forall\, n = 1,\dots,N\}, \tag{2.1}$$

where

$$\mathbb{P}_k(I_n, \mathbb{V}) := \{v : I_n \to \mathbb{V} : \ v(t) = \sum_{j=0}^{k} w_j t^j \ \forall\, t \in I_n, w_j \in \mathbb{V} \ \forall\, j = 0, \ldots, k\}$$

denotes the space of $\mathbb{V}$-valued polynomials of order k in time. For functions $v \in \mathbb{P}_k^{\mathrm{dc}}$, we define the jump at t_n as follows:

$$v_n^- := \lim_{t \nearrow t_n} v(t), \quad v_n^+ := \lim_{t \searrow t_n} v(t), \quad [v]_n := v_n^+ - v_n^-,$$

where v_0^- is to be understood as $v(0) := v_0 \in L^2(\Omega)$, *i.e.* given initial data.

2.1. Discontinuous Galerkin methods

The discontinuous Galerkin method of order $k \geq 0$ (denoted by dG(k)) to approximate problem (1.4) reads [34]:

For given $u_0 \in L^2(\Omega)$ and $f \in L^2((0,T), L^2(\Omega))$, find $u_\tau \in \mathbb{P}_k^{\mathrm{dc}}$, such that with $u_{\tau_0}^- := u_0$ it holds

$$\int_{I_n} \left[(\partial_t u_\tau, v) + a(u_\tau, v)\right] \mathrm{d}t + ([u_\tau]_{n-1}, v_{n-1}^+) = \int_{I_n} (f, v) \ \mathrm{d}t \quad \forall\, v \in \mathbb{P}_k^{\mathrm{dc}}, \quad 1 \leq n \leq N \tag{2.2}$$

This method is known to be strongly A-stable (L-stable) (Chap. 4.1.3, Lem. 3 + 5 in [27]). While the method is convergent of order $k+1$ when measured in $\|\cdot\|_{L^2((0,T),\mathbb{V})}$, it is super convergent of order $2k+1$ when only considering the solution at the nodal values $u_\tau(t_n)$ (Thm. 12.3, p. 211 in [34]).

Notice that on each time interval I_n, $u_\tau|_{I_n}$ may be written as

$$u_\tau(t) = \sum_{j=1}^{k+1} u_n^j \varphi_{n,j}(t) \quad \forall\, t \in I_n, \tag{2.3}$$

when $\{\varphi_{n,j} \in \mathbb{P}_k(I_n, \mathbb{R}), j = 1, \ldots, k+1\}$ denotes a set of basis vectors for the space $\mathbb{P}_k(I_n, \mathbb{V})$ and $u_n^j \in \mathbb{V}$. Correspondingly, any test function $v_j \in \mathbb{P}_k^{\mathrm{dc}}$ can be decomposed on the time interval I_n as a sum of terms of the form:

$$v(x)\varphi(t), \tag{2.4}$$

where $v \in \mathbb{V}$ is constant in time and $\varphi \in \mathbb{P}_k(I_n, \mathbb{R})$.

In order to derive a practical method, a basis of $\mathbb{P}_k(I_n, \mathbb{V})$ has to be specified and the time integrals in equation (2.2) need to be replaced by numerical quadrature. To this end, let

$$\omega_{n,q} \in \mathbb{R}, \ t_{n,q} \in I_n, q = 1, \ldots, N_Q \tag{2.5}$$

denote quadrature weights and quadrature points, respectively, such that the resulting quadrature operator

$$\begin{aligned} \mathcal{Q}_n &: C^0(\bar{I}_n) \to \mathbb{R}, \\ \mathcal{Q}_n[p] &:= \sum_{q=1}^{N_Q} \omega_{n,q} p(t_{n,q}) \end{aligned} \tag{2.6}$$

is exact for $p \in \mathbb{P}_{2k}(I_n, \mathbb{R})$. In particular, the terms $(\partial_t u_\tau, v)$ and (u_τ, v) are integrated exactly if such a quadrature formula is used.

Replacing the integrals in equation (2.2) by numerical quadrature and using a (yet to be specified) basis $\{\varphi_{n,j}, j = 1, \ldots, k+1\}$ of $\mathbb{P}_k(I_n, \mathbb{V})$ leads to the following time-discrete formulation of our problem on each time interval I_n:

Let $n \in \{1, \ldots N\}$. For given $u_\tau|_{I_{n-1}}$ from the previous time step (or initial data if $n = 1$), find $k+1$ coefficients $u_n^1, \ldots, u_n^{k+1} \in \mathbb{V}$ such that the following equations hold for all $v \in \mathbb{V}$:

$$\begin{aligned} \sum_{j=1}^{k+1} \gamma_{i,j}(u_n^j, v) + \alpha_{i,j} a(u_n^j, v) = \\ (u_{n-1}^-, v)\varphi_{n,i}(t_{n-1}) + \mathcal{Q}_n[\langle f(t), v\rangle \varphi_{n,i}(t)] \qquad \text{for } 1 \le i \le k+1, \end{aligned} \tag{2.7}$$

$$\text{where} \quad \gamma_{i,j} := \mathcal{Q}_n[\varphi'_{n,j}(t)\varphi_{n,i}(t)] + \varphi_{n,j}(t_{n-1})\varphi_{n,i}(t_{n-1}) \quad \text{and} \quad \alpha_{i,j} := \mathcal{Q}_n[\varphi_{n,j}(t)\varphi_{n,i}(t)].$$

Remarks:

- At this point, it is important to note that by definition the coefficients $\gamma_{i,j}$ and $\alpha_{i,j}$ are computed exactly, due to the assumption that $\mathcal{Q}_n$ is exact for $p \in \mathbb{P}_{2k}(I_n, \mathbb{R})$.
- Transformation to a fixed reference interval $\hat{I} = (0, 1)$ and using standard properties of basis functions and their corresponding reference basis functions $\{\hat{\varphi}_i\}$ on $\hat{I}$ yields

$$\begin{aligned} \gamma_{i,j} &= \mathcal{Q}_n[\varphi'_{n,j}(t)\varphi_{n,i}(t)] + \varphi_{n,j}(t_{n-1})\varphi_{n,i}(t_{n-1}) = \int_{I_n} \varphi'_{n,j}(t)\varphi_{n,i}(t)\, \mathrm{d}t + \varphi_{n,j}(t_{n-1})\varphi_{n,i}(t_{n-1}) \\ &= \int_0^1 \hat{\varphi}'_j(\hat{t})\hat{\varphi}_i(\hat{t})\, \mathrm{d}\hat{t} + \hat{\varphi}_j(0)\hat{\varphi}_i(0) =: \hat{\gamma}_{i,j}, \text{ and} \\ \alpha_{i,j} &= \mathcal{Q}_n[\varphi_{n,j}(t)\varphi_{n,i}(t)] = \int_{I_n} \varphi_j(t)\varphi_i(t)\, \mathrm{d}t = \tau_n \int_0^1 \hat{\varphi}_j(\hat{t})\hat{\varphi}_i(\hat{t})\, \mathrm{d}\hat{t} =: \tau_n \hat{\alpha}_{i,j} \end{aligned} \tag{2.8}$$

  for coefficients $\hat{\gamma}_{i,j}$ and $\hat{\alpha}_{i,j}$ which depend on the particular choice of basis functions, but not on τ_n.
- Up to now, a basis for the space $\mathbb{P}_k(I_n, \mathbb{V})$ has not been specified yet, the reason being that by construction our preconditioner will be insensitive to the particular choice of basis functions (see the remarks for Lem. 3.2). Following [27], in this paper we choose $\mathcal{Q}_n$ to be the $(k+1)$-point right-sided Gauß–Radau formula on $I_n = (t_{n-1}, t_n]$, which is exact for $p \in \mathbb{P}_{2k}$. With this choice, one of the quadrature points is given by t_n (denoting $t_{n,k+1} := t_n$ for the sake of simplicity). Correspondingly, we select $k+1$ Lagrange basis functions $\varphi_{n,j}$ fulfilling

$$\varphi_{n,j}(t_{n,i}) = \delta_{ij}, \quad 1 \le i, j \le k+1 \tag{2.9}$$

  at the time quadrature nodes $t_{n,i} \in I_n, i = 1, \ldots, k+1$. Since $t_n = t_{n,k+1}$, this leads to the property that $u_\tau(t_n) = u_n^{k+1}$.

2.2. Space discretization and fully discrete problem

In order to derive a fully discrete scheme let us denote by $\mathbb{V}_h \subset \mathbb{V}$ a conforming finite element space. On $\mathbb{V}_h$, we introduce the operator $A_h : \mathbb{V}_h \to \mathbb{V}_h$ and the identity operator $I_h : \mathbb{V}_h \to \mathbb{V}_h$ by:

$$(A_h u_h, v_h) = a(u_h, v_h) \quad \forall\, u_h, v_h \in \mathbb{V}_h, \tag{2.10}$$

$$(I_h u_h, v_h) = (u_h, v_h) \quad \forall\, u_h, v_h \in \mathbb{V}_h. \tag{2.11}$$

On each time interval I_n, we can now define our fully discrete problem based on equation (2.7) as a matrix valued problem on $\mathbb{V}_h$. To this end we introduce the operator-valued system matrices

$$\mathbf{G}_h := \begin{pmatrix} \gamma_{1,1} I_h & \ldots & \gamma_{1,k+1} I_h \\ \vdots & \ddots & \vdots \\ \gamma_{k+1,1} I_h & \ldots & \gamma_{k+1,k+1} I_h \end{pmatrix}, \quad \mathbf{B}_h := \begin{pmatrix} \alpha_{1,1} A_h & \ldots & \alpha_{1,k+1} A_h \\ \vdots & \ddots & \vdots \\ \alpha_{k+1,1} A_h & \ldots & \alpha_{k+1,k+1} A_h \end{pmatrix} \tag{2.12}$$

and the right-hand side vector

$$\mathbf{f}_h := \begin{pmatrix} (u_{n-1}^-, v)\varphi_{n,1}(t_{n-1}) + \mathcal{Q}_n[\langle f(t), v\rangle \varphi_{n,1}(t)] \\ \vdots \\ (u_{n-1}^-, v)\varphi_{n,k+1}(t_{n-1}) + \mathcal{Q}_n[\langle f(t), v\rangle \varphi_{n,k+1}(t)] \end{pmatrix}. \tag{2.13}$$

The fully discrete solution $\mathbf{u}_h = \left(u_h^1, \ldots, u_h^{k+1}\right)^T \in (\mathbb{V}_h)^{k+1}$ on I_n can now be computed from solving the following problem:

Let $u_\tau|_{I_{n-1}}$ be given from the previous time step or initial data, find $\boldsymbol{u}_h \in (\mathbb{V}_h)^{k+1}$ such that

$$(\mathbf{G}_h + \mathbf{B}_h)\,\mathbf{u}_h = \mathbf{f}_h. \tag{2.14}$$

3. An efficient solution strategy for the coupled system

This section is concerned with the main objective of this paper, namely with deriving an efficient solution strategy for the coupled problem (2.14).

3.1. Transformation to block diagonal form

In what follows, let the following tensor product notation hold for $\mathbf{C} \in \mathbb{R}^{r\times r}$, $r \in \mathbb{N}$ and an operator L on $\mathbb{V}_h$:

$$\mathbf{C} \otimes L := \begin{pmatrix} \mathbf{C}_{1,1}L & \ldots & \mathbf{C}_{1,r}L \\ \vdots & \ddots & \vdots \\ \mathbf{C}_{r,1}L & \ldots & \mathbf{C}_{r,r}L \end{pmatrix}. \tag{3.1}$$

Using this notation, problem (2.14) can be recast in the form

$$(\mathbf{g} \otimes I_h + \tau_n \mathbf{b} \otimes A_h)\,\mathbf{u}_h = \mathbf{f}_h, \tag{3.2}$$

where the coefficient matrices $\mathbf{g}, \mathbf{b} \in \mathbb{R}^{(k+1)\times(k+1)}$ are defined as $\mathbf{g}_{i,j} := \hat{\gamma}_{i,j}$ and $\mathbf{b}_{i,j} := \frac{1}{\tau_n}\alpha_{i,j} = \hat{\alpha}_{i,j}$, with $\hat{\alpha}, \hat{\gamma}$ as in equation (2.8). Note that by definition $\mathbf{b}$ and $\mathbf{g}$ do not depend on τ_n, but only on the choice of the basis.

Since $\mathbf{b}$ is a mass matrix built from linearly independent functions, it is invertible. Then multiplying equation (3.2) by $\mathbf{b}^{-1}$ yields

$$\left(\mathbf{b}^{-1}\mathbf{g} \otimes I_h + \tau_n \mathbf{1} \otimes A_h\right)\mathbf{u}_h = \mathbf{b}^{-1}\mathbf{f}_h, \tag{3.3}$$

where $\mathbf{1}$ denotes the $r \times r$ identity matrix. One key idea, well-known in the Runge–Kutta community (see for instance [13], Chap. IV.8) due to the pioneering work by Butcher [6], is to transform the matrix $\mathbf{b}^{-1}\mathbf{g}$ into a matrix of simple form, such as diagonal or block diagonal. To this end, for the rest of this paper we make the following assumption:

Assumption 3.1. Assume that the matrix $\mathbf{b}^{-1}\mathbf{g}$ is diagonalizable (over $\mathbb{C}$).

Remarks:

- The authors failed to prove that Assumption 3.1 is fulfilled for all $k \in \mathbb{N}$. However, computational tests clearly showed that the assumption is fulfilled for all $k \leq 50$, which is far beyond practical needs. A similar result was reported in [28], see remarks on Method (ii), Section 6.3, and the question of diagonalizability of the matrix $\mathbf{b}^{-1}\mathbf{g}$ seems to remain open.
- Note that Assumption 3.1 is not really needed. Our strategy outlined below would also apply to the general situation. In this case one would have to replace transformation equation (3.10) by a real Schur transformation, *i.e.* a transformation into a block upper triangular form, see [12, 28], Method (i) in Section 6.3 for more details. Of course it is more convenient and more effective to work with a block diagonal structure.

For convenience the eigenvalues $\lambda_1, \ldots, \lambda_r, \lambda_{r+1}, \ldots, \lambda_{k+1}$ of $\mathbf{b}^{-1}\mathbf{g}$ may be ordered such that the first r eigenvalues $\lambda_1, \ldots, \lambda_r$ are real, while the remaining $k-r+1$ eigenvalues are complex. Since these eigenvalues appear as complex conjugates, for the sake of notation we let $\{\lambda_{r+1}, \ldots, \lambda_{k+1}\} = \{\lambda_{r+1}, \overline{\lambda}_{r+1}, \ldots, \lambda_n, \overline{\lambda}_n\}$ for $n = \frac{k-r+1}{2}$, where $\Im(\lambda_i) > 0$ and $\Im(\overline{\lambda}_i) < 0$ for $r+1 \leq i \leq n$.

Denoting the matrix holding the (possibly complex valued) eigenvectors of $\mathbf{b}^{-1}\mathbf{g}$ which correspond to the eigenvalues $\{\lambda_1, \ldots, \lambda_r, \lambda_{r+1}, \overline{\lambda}_{r+1}, \ldots, \lambda_n, \overline{\lambda}_n\}$ by $\tilde{\mathbf{V}} := \begin{bmatrix}\mathbf{x}_1 \ldots \mathbf{x}_r \ \mathbf{x}_{r+1} \ \overline{\mathbf{x}}_{r+1} \ldots \mathbf{x}_n \ \overline{\mathbf{x}}_n\end{bmatrix} \in \mathbb{C}^{(k+1)\times(k+1)}$ and letting $\tilde{\mathbf{D}} := \operatorname{diag}(\lambda_1, \ldots, \lambda_r, \lambda_{r+1}, \overline{\lambda}_{r+1}, \ldots, \lambda_n, \overline{\lambda}_n) \in \mathbb{C}^{(k+1)\times(k+1)}$ Assumption 3.1 yields

$$\left(\mathbf{b}^{-1}\mathbf{g}\right)\tilde{\mathbf{V}} = \tilde{\mathbf{V}}\tilde{\mathbf{D}}. \tag{3.4}$$

Remark: At this point, it should be emphasized that if one is willing to use complex arithmetic, the above transformation applied to equation (3.3) would directly lead to solving the block diagonal but complex-valued system

$$\left(\tilde{\mathbf{D}} \otimes I_h + \tau_n \mathbf{1} \otimes A_h\right)\tilde{\mathbf{u}}_h = \tilde{\mathbf{V}}^{-1}\mathbf{b}^{-1}\mathbf{f}_h, \tag{3.5}$$

where $\tilde{\mathbf{u}}_h = \tilde{\mathbf{V}}^{-1}\mathbf{u}_h$. The special structure of the decoupled problems in the system above then allows for an efficient treatment in terms of iterative solvers, see for instance [10,11]. As stated in the introduction, one of our design principles was to avoid complex arithmetic and we therefore make use of a real-valued block diagonal transformation. To this end let us define the block diagonal matrix

$$\tilde{\mathbf{T}} := \begin{pmatrix} \mathbf{1} & & & \\ & \tilde{\mathbf{R}} & & \\ & & \ddots & \\ & & & \tilde{\mathbf{R}} \end{pmatrix} \in \mathbb{C}^{(k+1)\times(k+1)} \quad \text{where} \quad \tilde{\mathbf{R}} := \frac{1}{2}\begin{pmatrix} 1 & -i \\ 1 & i \end{pmatrix}. \tag{3.6}$$

Using this matrix one easily checks that

$$\tilde{\mathbf{T}}^{-1}\tilde{\mathbf{D}}\tilde{\mathbf{T}} = \begin{pmatrix} \Lambda_1 & & \\ & \ddots & \\ & & \Lambda_n \end{pmatrix} \in \mathbb{R}^{(k+1)\times(k+1)} \tag{3.7}$$

is block diagonal and real valued with $\Lambda_i = \lambda_i$ for $i \leq r$ and the 2×2 blocks

$$\Lambda_i = \begin{pmatrix} \alpha_i & \beta_i \\ -\beta_i & \alpha_i \end{pmatrix}, \tag{3.8}$$

where $\alpha_i = \Re(\lambda_i)$ and $\beta_i = \Im(\lambda_i)$ for $r+1 \leq i \leq n$. Furthermore, we get that

$$\tilde{\mathbf{V}}\tilde{\mathbf{T}} = \begin{bmatrix}\mathbf{x}_1 \ldots \mathbf{x}_r \ \Re(\mathbf{x}_{r+1}) \ \Im(\mathbf{x}_{r+1}) \ldots \Re(\mathbf{x}_n) \ \Im(\mathbf{x}_n)\end{bmatrix} \in \mathbb{R}^{(k+1)\times(k+1)}. \tag{3.9}$$

As a result, we have constructed a real valued transformation matrix $\mathbf{V} := \tilde{\mathbf{V}}\tilde{\mathbf{T}} \in \mathbb{R}^{(k+1)\times(k+1)}$ such that

$$\mathbf{V}^{-1}\left(\mathbf{b}^{-1}\mathbf{g}\right)\mathbf{V} = \begin{pmatrix} \Lambda_1 & & \\ & \ddots & \\ & & \Lambda_n \end{pmatrix}. \tag{3.10}$$

Introducing the transformed solution variable $\mathbf{w}_h = \mathbf{V}^{-1}\mathbf{u}_h$ and multiplying (3.3) by $\mathbf{V}^{-1}$, leads to the equivalent formulation

$$\mathbf{S}_h\mathbf{w}_h = \mathbf{V}^{-1}\mathbf{b}^{-1}\mathbf{f}_h, \tag{3.11}$$

TABLE 1. Eigenvalues of the matrix $\mathbf{b}^{-1}\mathbf{g}$ for the dG(k) methods, $k = 1, \dots, 4$.

dG(1)	dG(2)	dG(3)	dG(4)
$2.0000 + 1.4142i$,	$2.6811 + 3.0504i$,	$3.2128 + 4.7731i$,	$3.6557 + 6.5437i$,
$2.0000 - 1.4142i$	$2.6811 - 3.0504i$,	$3.2128 - 4.7731i$,	$3.6557 - 6.5437i$,
	3.6378	$4.7872 + 1.5675i$,	$5.7010 + 3.2103i$,
		$4.7872 - 1.5675i$	$5.7010 - 3.2103i$,
			6.2867

where

$$\mathbf{S}_h = \begin{pmatrix} \lambda_1 I_h + \tau_n A_h & & & & & & & & \\ & \ddots & & & & & & & \\ & & \lambda_r I_h + \tau_n A_h & & & & & & \\ & & & \alpha_{r+1} I_h + \tau_n A_h & \beta_{r+1} I_h & & & & \\ & & & -\beta_{r+1} I_h & \alpha_{r+1} I_h + \tau_n A_h & & & & \\ & & & & & \ddots & & & \\ & & & & & & \alpha_n I_h + \tau_n A_h & \beta_n I_h & \\ & & & & & & -\beta_n I_h & \alpha_n I_h + \tau_n A_h \end{pmatrix}. \quad (3.12)$$

We are thus led to the problem of solving a block diagonal system $\mathbf{S}_h$ which reduces to solving r standard 1×1 implicit Euler-like problems, and coupled 2×2 block systems of the form

$$\begin{pmatrix} \alpha I_h + \tau_n A_h & \beta I_h \\ -\beta I_h & \alpha I_h + \tau_n A_h \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}, \quad (3.13)$$

whose efficient solution will be subject of the next subsection. Note that due to the block diagonal structure, the solution of the sub-problems (3.13) can be done in parallel.

Remark: Table 1 shows the eigenvalues of the matrix $\mathbf{b}^{-1}\mathbf{g}$ for the dG(k) methods with $k = 1, \dots, 4$. Notice that all eigenvalues listed in Table 1 have positive real part, and it will turn out that the sign of the real part α of the eigenvalues is a crucial point in our considerations. We thus provide the following lemma.

Lemma 3.2. *Let λ be an eigenvalue of the matrix $\boldsymbol{b}^{-1}\boldsymbol{g}$. Then it holds*

$$\Re(\lambda) \geq 0.$$

Proof. By definition of $\mathbf{b}$ and $\mathbf{g}$, λ is an eigenvalue iff there exists $v \in \mathbb{P}_k([0,1], \mathbb{C})$ with $\int_0^1 |v(\hat{t})|^2 \mathrm{d}\hat{t} = 1$ and

$$\int_0^1 v'(\hat{t})\bar{w}(\hat{t})\mathrm{d}\hat{t} + (v\bar{w})(0) = \lambda \int_0^1 v(\hat{t})\bar{w}(\hat{t})\mathrm{d}\hat{t}$$

for all $w \in \mathbb{P}_k([0,1], \mathbb{C})$. Here, $\bar{w}$ denotes the complex conjugate of w. Integrating by parts yields

$$\int_0^1 v'(\hat{t})\bar{w}(\hat{t})\mathrm{d}\hat{t} + (v\bar{w})(0) = -\int_0^1 v(\hat{t})\bar{w}'(\hat{t})\mathrm{d}\hat{t} + (v\bar{w})(1).$$

Adding the left-hand side and the complex conjugate of the right-hand side gives

$$\int_0^1 v'(\hat{t})\bar{w}(\hat{t}) - \bar{v}(\hat{t})w'(\hat{t})\mathrm{d}\hat{t} \; + \left[(v\bar{w})(0) + (\bar{v}w)(1)\right] = \lambda \int_0^1 v(\hat{t})\bar{w}(\hat{t})\mathrm{d}\hat{t} \; + \; \bar{\lambda} \int_0^1 \bar{v}(\hat{t})w(\hat{t})\mathrm{d}\hat{t}.$$

Setting $w := v$ yields

$$0 \leq |v(0)|^2 + |v(1)|^2 = 2\,\Re(\lambda). \qquad \square$$

As already mentioned earlier, the proof of the above lemma also showed that the eigenvalues of $\mathbf{b}^{-1}\mathbf{g}$ are independent of the specific choice of the basis for $\mathbb{P}_k(I_n, \mathbb{V})$.

3.2. Solving the 2 × 2 block system by a preconditioned Schur complement formulation

The solution of the 2×2 block system (3.13) can be achieved in numerous ways. In this paper, we will make use of a Schur complement formulation: by block elimination, first w_2 is obtained by solving

$$S_h w_2 = \beta f_1 + (\alpha I_h + \tau_n A_h) f_2, \text{ where } S_h = (\alpha I_h + \tau_n A_h)^2 + \beta^2 I_h, \tag{3.14}$$

and then w_1 is determined from

$$I_h w_1 = \frac{1}{\beta} \left[(\alpha I_h + \tau_n A_h) - f_2 \right]. \tag{3.15}$$

Clearly, the performance of the entire process will depend on how efficiently the Schur complement equation (3.14) can be solved, and some remarks seem to be in order.

Remarks:

- The Schur complement operator S_h is a (discretized) fourth order differential operator, which means that the conditioning of problem (3.14) will be, in general, much worse than that of problem (3.13). To be more precise, for uniformly elliptic second order operators A_h and H^1 conforming Lagrange finite elements the condition number of A_h in the worst case (*i.e.* for fixed time step size τ but variable mesh size h) is bounded by
$$\kappa(A_h) \leq C h^{-2} \quad \text{and hence} \quad \kappa(S_h) \leq C h^{-4},$$
see for instance Theorem 9.11 in [8].
- Due to the previous remark, an efficient solution strategy for the Schur complement formulation (3.14) using iterative solvers will only be possible with a suitable preconditioner that will be constructed in this section.
- If A_h is symmetric positive definite, then also the Schur complement operator S_h is symmetric positive definite, and solvers exploiting this fact can be used (for instance, CG). It is worth pointing out that the block system (3.13) does not possess this property since it has saddle point structure.

The preconditioner for (3.14) is based on ideas developed for a class of fourth order problems presented and analyzed in [3]. The main ingredient is an inexact factorization of the Schur complement operator S_h, for which uniform bounds can be proved. A corresponding preconditioner for the special case of dG(1) and cGP(2) time discretizations was already presented in [36]. Note that an exact factorization for S_h is also possible, but leads to complex-valued equations, as pointed out in [26].

The preconditioner will be derived from and analyzed under the assumption that A_h is symmetric, positive definite. We will later demonstrate with a numerical example that the proposed preconditioner also works for more general operators. Consider the symmetric left-right preconditioned operator

$$P_h(\mu) := (\mu I_h + \tau_n A_h)^{-1} S_h (\mu I_h + \tau_n A_h)^{-1}, \tag{3.16}$$

where $\mu > 0$ denotes a yet to be specified parameter. Note that the spectrum of $\mu I_h + \tau_n A_h$ is given by

$$\sigma(\mu I_h + \tau_n A_h) = \{\mu + \xi : 0 \leq \xi \in \sigma(\tau_n A_h)\} \subset (\mu, \infty). \tag{3.17}$$

Therefore, for fixed $\xi \in \sigma(\tau_n A_h)$, the corresponding eigenvalue $\tilde{\xi}_\mu(\xi)$ of the preconditioned operator $P_h(\mu)$ reads:

$$\tilde{\xi}_\mu(\xi) = \frac{(\alpha + \xi)^2 + \beta^2}{(\mu + \xi)^2}. \tag{3.18}$$

In order to highlight the role of the parameter μ let us first assume that α is strictly positive and $\mu = \alpha > 0$. Then

$$\tilde{\xi}_\alpha(\xi) = 1 + \frac{\beta^2}{(\alpha + \xi)^2} \tag{3.19}$$

and since $\tilde{\xi}$ is monotonically decreasing in ξ, we have

$$\sup_{\xi\in\sigma(\tau_n A_h)} \tilde{\xi}_\alpha(\xi) \leq \tilde{\xi}_\alpha(0) = 1 + \frac{\beta^2}{\alpha^2}, \tag{3.20}$$

$$\inf_{\xi\in\sigma(\tau_n A_h)} \tilde{\xi}_\alpha(\xi) \geq \lim_{\xi\to\infty} \tilde{\xi}_\alpha(\xi) = 1 \tag{3.21}$$

and consequently $\sigma(P_h(\alpha)) \subset (1, 1+\frac{\beta^2}{\alpha^2})$. Hence, the condition number of the preconditioned operator $P_h(\alpha)$ is bounded by

$$\kappa(P_h(\alpha)) \leq 1 + \frac{\beta^2}{\alpha^2} \quad \text{if} \quad \mu = \alpha > 0. \tag{3.22}$$

For the special case that $\alpha = 0$ (which cannot be excluded *a priori* by the result of Lem. 3.2), choosing the parameter $\mu = \beta$ in (3.18) leads to

$$\tilde{\xi}_\beta(\xi) = \frac{\xi^2+\beta^2}{(\beta+\xi)^2} \quad \text{with} \quad \frac{1}{2} \leq \tilde{\xi}_\beta(\xi) \leq 1 \quad \text{and therefore} \quad \kappa(P_h(\beta)) \leq 2. \tag{3.23}$$

Note that both of the above bounds for the condition numbers are already insensitive to spatial discretization (*e.g.* mesh size) and time step size parameters. However, α and β still depend on the temporal basis. Experimental results show (see Fig. 1) that for increasing polynomial degree k, the ratio $\frac{\beta^2}{\alpha^2}$ in (3.22) increases as well. Consequently, we aim at finding a parameter μ which resolves this dependency and leads to uniform bounds.

Recalling that $\lambda = \alpha + i\beta$, observe that the Schur complement operator S_h admits the following decomposition,

$$S_h = (\alpha I_h + \tau_n A_h)^2 + \beta^2 I_h = (\tau_n A_h + \lambda I_h)(\tau_n A_h + \bar{\lambda} I_h).$$

A suitable candidate the for the real parameter μ in (3.16) is thus given by $\mu = |\lambda| = \sqrt{\alpha^2+\beta^2}$. In the following Proposition, we show that indeed the optimal parameter μ_{opt} is specified by the above relation, and prove uniform bounds for the resulting left-right preconditioned operator.

Proposition 3.3. *Let A_h be symmetric and positive definite. For $\alpha \geq 0, \beta \in \mathbb{R}$ given, let the parameter $\mu_{\text{opt}} := \sqrt{\alpha^2+\beta^2}$. Then the preconditioned operator $P_h(\mu_{\text{opt}})$ has condition number*

$$\kappa(P_h(\mu_{\text{opt}})) \leq 2 - 2\frac{\alpha}{\beta^2}\left(\sqrt{\alpha^2+\beta^2} - \alpha\right) \leq 2. \tag{3.24}$$

Remark: For the dG(1) method, in its corresponding 2×2 system (3.13) we have that $\alpha = 2$ and $\beta = \sqrt{2}$, see Table 1. Proposition 3.3 then implies that the condition number of the preconditioned Schur complement operator is bounded by

$$\kappa(P_h(\mu_{\text{opt}})) \leq 6 - 2\sqrt{6} \approx 1.101.$$

Proof. In the proof we assume that $\alpha > 0$ because we have already derived that the parameter $\mu = \beta$ leads exactly to the uniform bounds specified in (3.24) for the case $\alpha = 0$. We first note that

$$\frac{\mathrm{d}}{\mathrm{d}\xi}\tilde{\xi}_\mu(\xi) = \frac{2}{(\mu+\xi)^3}\left[(\alpha+\xi)(\mu-\alpha) - \beta^2\right], \tag{3.25}$$

which means that $\tilde{\xi}_\mu$ is monotonically decreasing in ξ if $\mu \leq \alpha$. Therefore, in this case, $\kappa(P_h(\mu)) \leq \frac{\tilde{\xi}_\mu(0)}{\lim_{\xi\to\infty}\tilde{\xi}_\mu(\xi)} \leq \frac{\tilde{\xi}_\mu(0)}{1} = \frac{\alpha^2+\beta^2}{\mu^2}$, which is minimal if $\mu = \alpha$ and would result in the non optimal bound (3.22).

For $\mu \geq \alpha$ we are interested in extremal points of $\tilde{\xi}_\mu(\xi)$. It is easy to check that

$$\frac{\mathrm{d}}{\mathrm{d}\xi}\tilde{\xi}_\mu(\xi^*) = 0 \Leftrightarrow \xi^* = \frac{\beta^2}{\mu - \alpha} - \alpha, \text{ and} \tag{3.26}$$

$$\xi^* \geq 0 \Leftrightarrow \mu \leq \alpha + \frac{\beta^2}{\alpha}, \tag{3.27}$$

and, since A_h was assumed to be positive definite, we need to check the behavior of $\tilde{\xi}_\mu(\xi)$ for $\alpha \leq \mu \leq \alpha + \frac{\beta^2}{\alpha}$. In this case, we have

$$\tilde{\xi}_\mu(0) = \frac{\alpha^2 + \beta^2}{\mu^2}, \quad \lim_{\xi\to\infty} \tilde{\xi}_\mu(\xi) = 1 \quad \text{and } \tilde{\xi}_\mu(\xi^*) = \frac{\beta^2}{\beta^2 + (\mu - \alpha)^2} \leq 1. \tag{3.28}$$

Direct computation shows that $\tilde{\xi}_\mu(0) \geq \tilde{\xi}_\mu(\xi^*)$ and therefore the condition number of $P_h(\mu)$ can be estimated by

$$\kappa(P_h(\mu)) \leq \frac{\max(1, \tilde{\xi}_\mu(0))}{\tilde{\xi}_\mu(\xi^*)}. \tag{3.29}$$

We need to distinguish between two cases: $\tilde{\xi}_\mu(0) \leq 1$ and $\tilde{\xi}_\mu(0) > 1$. The first case requires $\mu^2 \geq \alpha^2 + \beta^2$ and yields

$$\kappa(P_h(\mu)) \leq \frac{1}{\tilde{\xi}_\mu(\xi^*)} = 1 + \frac{(\mu - \alpha)^2}{\beta^2}. \tag{3.30}$$

For the second case,

$$\kappa(P_h(\mu)) \leq \frac{\tilde{\xi}_\mu(0)}{\tilde{\xi}_\mu(\xi^*)} = \frac{\alpha^2 + \beta^2}{\beta^2}\frac{\beta^2 + (\mu - \alpha)^2}{\mu^2}, \tag{3.31}$$

which is monotonically decreasing in μ for $\mu^2 < \alpha^2 + \beta^2$. Both cases (3.30) and (3.31) yield that $\kappa(P_h(\mu))$ is minimized when

$$\mu = \mu_{\mathrm{opt}} := \sqrt{\alpha^2 + \beta^2}, \tag{3.32}$$

which finally gives

$$\kappa(P_h(\mu_{\mathrm{opt}})) \leq 1 + \frac{(\mu_{\mathrm{opt}} - \alpha)^2}{\beta^2} = 2 - 2\frac{\alpha}{\beta^2}\left(\sqrt{\alpha^2 + \beta^2} - \alpha\right) \tag{3.33}$$

and proves Proposition 3.3. The remaining case $\mu \geq \alpha + \frac{\beta^2}{\alpha}$ leads to non optimal bounds, since from (3.25) one concludes that $\tilde{\xi}_\mu$ is monotonically increasing in ξ, and therefore

$$\kappa(P_h(\mu)) \leq \frac{\lim_{\xi\to\infty} \tilde{\xi}_\mu(\xi)}{\tilde{\xi}_\mu(0)} = \frac{\mu^2}{\alpha^2 + \beta^2},$$

while $\frac{\mu^2}{\alpha^2+\beta^2} \geq 1 + \frac{\beta^2}{\alpha^2}$. □

For the sake of brevity, we will always write μ_{opt} to refer to the optimal value $\mu_{\mathrm{opt}} = \sqrt{\alpha^2 + \beta^2}$ for each block in (3.11), and will not distinguish between μ_{opt} for different blocks, time discretization methods or temporal polynomial degrees k.

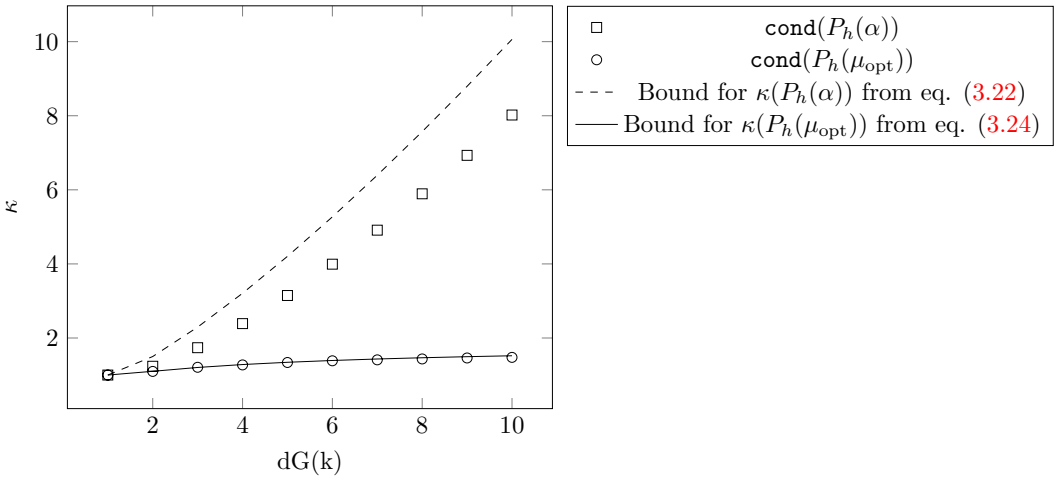

FIGURE 1. Condition numbers of the preconditioned operators $P_h(\alpha)$ and $P_h(\mu)$ for increasing polynomial degree k together with their theoretical bounds given in (3.22), (3.24).

This subsection is finished by a numerical experiment, indicating that the bounds (3.22) and (3.24) are sharp in the sense that $\kappa(P_h(\mu))$ is unbounded in the polynomial degree k for $\mu = \alpha$ but remains bounded for $\mu = \mu_{\text{opt}}$. To this end, consider dG(k) and compute the eigenvalues λ_i of the corresponding coefficient matrix $\mathbf{b}^{-1}\mathbf{g}$. The maximum condition numbers of the preconditioned Schur complement operators $P_h(\alpha)$ and $P_h(\mu_{\text{opt}})$ corresponding to each block are evaluated both from *a priori* estimates (3.22), (3.24), and from a fully discretized 1d numerical example (*i.e.*, a realistic discrete operator $\tau_n A_h$ corresponding to certain mesh size and time step size parameters, see Sect. 5.3.1 for details). The latter is done by using the MATLAB function `cond()`. Results are depicted in Figure 1. Clearly, increasing the temporal polynomial degree k leads to an increase of the condition number of $P_h(\alpha)$, while it stays bounded for $P_h(\mu_{\text{opt}})$ indicating the optimality of the preconditioner with respect to all parameters.

4. Numerical realization

In this section we elaborate on the choice of a practical finite element space and comment on our implementation.

4.1. Choice of finite element spaces

Note that except for conformity ($\mathbb{V}_h \subset \mathbb{V}$) so far no special restriction on the finite element space was assumed. This makes our preconditioner suitable for a broad class of finite element discretizations.

In our numerical realization, a standard polynomial ansatz on simplicial elements is chosen. To this end, let $\mathcal{T}_h$ be a conforming triangulation of Ω, and define the space of continuous, piecewise polynomial functions on Ω:

$$\mathbb{V}_h = \mathbb{V}_{h,m} := \{v_h \in C^0(\Omega) : \ v_{h|T} \in \mathbb{P}_m(T, \mathbb{R}) \ \forall T \in \mathcal{T}_h\}. \tag{4.1}$$

For a basis $\varphi_i, i = 1, \ldots, n_{\text{dof}}$ of $\mathbb{V}_h$, define the mass matrix, stiffness matrix, and right-hand sides by

$$\mathbf{M}_{ij} = (\varphi_j, \varphi_i) \qquad i, j = 1, \ldots, n_{\text{dof}} \tag{4.2}$$

$$\mathbf{A}_{ij} = a(\varphi_j, \varphi_i) \qquad i, j = 1, \ldots, n_{\text{dof}} \tag{4.3}$$

$$\mathbf{f}_{1i} = (f_1, \varphi_i) \qquad i = 1, \ldots, n_{\text{dof}} \tag{4.4}$$

$$\mathbf{f}_{2i} = (f_2, \varphi_i) \qquad i = 1, \ldots, n_{\text{dof}}, \tag{4.5}$$

respectively, where f_1 and f_2 correspond to the right-hand side terms from the block system (3.13). Furthermore, for a parameter $\theta > 0$, define the matrix

$$\mathbf{A}_\theta = \theta\mathbf{M} + \tau_n\mathbf{A}. \tag{4.6}$$

With the above defined matrices $\mathbf{M}, \mathbf{A}, \mathbf{A}_\theta$ and vectors $\mathbf{f}, \mathbf{g}$ it holds:

$$(I_h u_h, \varphi_l) = (u_h, \varphi_l) = (\mathbf{M}\mathbf{u})_l, \tag{4.7}$$
$$(A_h u_h, \varphi_l) = a(u_h, \varphi_l) = (\mathbf{A}\mathbf{u})_l, \tag{4.8}$$
$$((\theta I_h + \tau_n A_h)u_h, \varphi_l) = \theta(u_h, \varphi_l) + \tau_n a(u_h, \varphi_l) = (\mathbf{A}_\theta\mathbf{u})_l, \tag{4.9}$$
$$(f_1, \varphi_l) = \mathbf{f}_{1l}, \tag{4.10}$$
$$(f_2, \varphi_l) = \mathbf{f}_{2l} \tag{4.11}$$

for $l = 1, \ldots, n_{\mathrm{dof}}$, where $\mathbf{u} \in \mathbb{R}^{n_{\mathrm{dof}}}, \mathbf{u} = (\mathbf{u}_i)_{i=1,\ldots,n_{\mathrm{dof}}}$ is the coefficient vector of $u_h \in \mathbb{V}_h$.

To derive the system of equations that corresponds to the operator equation (3.14) in $\mathbb{V}_h$ we explain at first the isomorphism $r_m : \mathcal{L}(\mathbb{V}_h, \mathbb{V}_h) \to \mathbb{R}^{N\times N}, N = n_{\mathrm{dof}}$, that assigns to a linear operator $A_h : \mathbb{V}_h \to \mathbb{V}_h$ its matrix representation $r_m(A_h) \in \mathbb{R}^{N\times N}$ in the following way (see [3]): Let $r_v : \mathbb{V}_h \to \mathbb{R}^N$ denote the finite element isomorphism that assigns to a function $u_h \in \mathbb{V}_h$ its nodal vector representation $\mathbf{u} = r_v(u_h) \in \mathbb{R}^N$. Then, the matrix representation $r_m(A_h) \in \mathbb{R}^{N\times N}$ is defined by the property

$$r_v(A_h u_h) = r_m(A_h) r_v(u_h) \qquad \forall\, u_h \in \mathbb{V}_h.$$

If we define as above the matrix $\mathbf{M} \in \mathbb{R}^{N\times N}$ and the matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ assigned to A_h by means of $\mathbf{A}_{kj} := (A_h\varphi_j, \varphi_k)$ for all $k, j = 1, \ldots, N$, we can show that $r_m(A_h) = \mathbf{M}^{-1}\mathbf{A}$. For two operators A_{1h}, A_{2h} with corresponding matrices $\mathbf{A}_1, \mathbf{A}_2$ we get $r_m(A_{1h}A_{2h}) = \mathbf{M}^{-1}\mathbf{A}_1\mathbf{M}^{-1}\mathbf{A}_2$. Now, using the fact that $r_v(f_2) = \mathbf{M}^{-1}\mathbf{f}_2$ and the analog for $r_v(f_1)$, we get that the operator equation (3.14) for $u_h \in \mathbb{V}_h$ is equivalent to the following matrix-vector equation for the nodal vector $\mathbf{w}_2 = r_v(w_2) \in \mathbb{R}^N$:

$$\left(\mathbf{M}^{-1}\mathbf{A}_\alpha\mathbf{M}^{-1}\mathbf{A}_\alpha + \beta^2\mathbf{I}\right)\mathbf{w}_2 = \beta\mathbf{M}^{-1}\mathbf{f}_1 + \mathbf{M}^{-1}\mathbf{A}_\alpha\mathbf{M}^{-1}\mathbf{f}_2 \tag{4.12}$$

or, by multiplying by $\mathbf{M}$ to save unnecessary inversions of $\mathbf{M}$,

$$\left(\mathbf{A}_\alpha\mathbf{M}^{-1}\mathbf{A}_\alpha + \beta^2\mathbf{M}\right)\mathbf{w}_2 = \beta\mathbf{f}_1 + \mathbf{A}_\alpha\mathbf{M}^{-1}\mathbf{f}_2 \tag{4.13}$$

Correspondingly, the left and right sided preconditioner components have the matrix representation

$$\mathbf{A}_{\mu_{\mathrm{opt}}} = \mu_{\mathrm{opt}}\mathbf{M} + \tau_n\mathbf{A}. \tag{4.14}$$

Note that an operator A_h has the same eigenvalues as its matrix representation $r_m(A_h)$ such that the results of Section 3 on the preconditioning can be applied to the iterative solution of the corresponding matrix-vector equation (4.13) directly.

In the case of a symmetric operator A_h, the solution of equation (4.13) can be done using the preconditioned Conjugate Gradient method (PCG). A pseudocode variant containing the operators from our setting is shown in Algorithm 1.

Once the essential unknown $\mathbf{w}_2$ was computed with Algorithm 1, the second unknown $\mathbf{w}_1$ can be obtained, in view of (3.15), from

$$\mathbf{M}\mathbf{w}_1 = \frac{1}{\beta}\left[\mathbf{A}_\alpha\mathbf{w}_2 - \mathbf{f}_2\right]. \tag{4.15}$$

Remark: The main numerical effort of Algorithm 1 lies in the application of the preconditioner $\mathbf{A}_{\mathrm{opt}}^{-1} = (\mu_{\mathrm{opt}}\mathbf{M} + \tau_n\mathbf{A})^{-1}$. This evaluation corresponds to solving one step of the implicit Euler method with time step size $\frac{\tau_n}{\mu_{\mathrm{opt}}}$. This problem is very well studied and can be solved efficiently with many methods, *e.g.* multigrid methods, Krylov subspace methods etc. Additionally, one inversion of the mass matrix $\mathbf{M}$ is necessary per application of the preconditioner, a problem that can be tackled efficiently by the aforementioned methods.

Algorithm 1. PCG for the Schur complement equation (4.13).

Require: Parameters α, β, right-hand side vectors $\mathbf{f}, \mathbf{g}$ and initial guess $\mathbf{x}_0$

$\omega_{\text{opt}} \leftarrow \sqrt{\alpha^2 + \beta^2}$ ▷ Compute optimal parameter for preconditioner
$\mathbf{x} \leftarrow \mathbf{x}_0$ ▷ Initial guess
$\mathbf{S} \leftarrow \mathbf{A}_\alpha \mathbf{M}^{-1} \mathbf{A}_\alpha + \beta^2 \mathbf{M}$ ▷ Define Schur complement operator
$\mathbf{A}_{\text{opt}} \leftarrow \omega_{\text{opt}} \mathbf{M} + \tau_n \mathbf{A}$ ▷ Define preconditioner
$\mathbf{b} \leftarrow \left(\beta \mathbf{f}_1 + \mathbf{A}_\alpha \mathbf{M}^{-1} \mathbf{f}_2\right)$ ▷ Compute right-hand side
$\mathbf{r}_0 \leftarrow \mathbf{r} \leftarrow \mathbf{S}\mathbf{x} - \mathbf{b}$ ▷ Compute initial residual
$\mathbf{h} \leftarrow \mathbf{A}_{\text{opt}}^{-1} \mathbf{M} \mathbf{A}_{\text{opt}}^{-1} \mathbf{r}_0$ ▷ Compute preconditioned residual
$\mathbf{d} \leftarrow \mathbf{h}$
loop
  $\beta \leftarrow \frac{1}{\langle \mathbf{r}, \mathbf{h} \rangle}$
  $\mathbf{z} \leftarrow \mathbf{S}\mathbf{d}$ ▷ Apply operator
  $\alpha \leftarrow \frac{\langle \mathbf{r}, \mathbf{h} \rangle}{\langle \mathbf{d}, \mathbf{z} \rangle}$
  $\mathbf{x} \leftarrow \mathbf{x} + \alpha \mathbf{d}$ ▷ Update solution
  $\mathbf{r} \leftarrow \mathbf{r} - \alpha \mathbf{z}$ ▷ Update residual
  return if $\|\mathbf{r}\| / \|\mathbf{r}_0\| < \text{tol}$
  $\mathbf{h} \leftarrow \mathbf{A}_{\text{opt}}^{-1} \mathbf{M} \mathbf{A}_{\text{opt}}^{-1} \mathbf{r}$ ▷ Compute preconditioned residual
  $\beta \leftarrow \beta \langle \mathbf{r}, \mathbf{h} \rangle$
  $\mathbf{d} \leftarrow \mathbf{h} + \beta \mathbf{d}$ ▷ Update search direction
end loop
return $\mathbf{x}$ ▷ $\mathbf{w}_2 \leftarrow \mathbf{x}$, solution to (4.13)

4.2. Implementation

The implementation was done using the MATLAB based finite element toolbox FELICITY [35]. FELICITY offers linear and quadratic finite elements on unstructured simplicial grids in 1, 2 and 3 space dimensions. To allow for fair comparisons and having full control over termination criteria of the linear solvers, we did not use the MATLAB built-in iterative solvers `pcg` and `bicg` to realize Algorithm 1, but relied on handcrafted versions instead.

The eigenvalues of the matrix $\mathbf{b}^{-1}\mathbf{g}$ and the transformation matrix $\mathbf{V}$ were directly obtained from MATLAB (only once at the beginning of each computation) and each Schur complement problem (3.14) was solved using Algorithm 1.

As termination criterion for Algorithm 1, we prescribed a relative tolerance $\text{tol} = 1e-10$. For the application of the preconditioner, as mentioned above, the main numerical effort lies in evaluating $\mathbf{A}_{\text{opt}}^{-1}$. An exact evaluation may turn out to be too costly in practical applications, and cheaper approximations should be considered. In our implementation, we allow for the following strategies to (approximatively) evaluate $\mathbf{A}_{\text{opt}}^{-1}$:

- An "exact" evaluation of $\mathbf{A}_{\text{opt}}^{-1}$ based on the MATLAB operator \.
- An approximation to $\mathbf{A}_{\text{opt}}^{-1}$ based on applying the algebraic multigrid solver AGMG [22–25] with different stopping tolerances. AGMG has shown excellent performance when applied to large scale discretizations of convection-diffusion equations.

For the inversion of the (well-conditioned) mass matrix which is needed in (4.15) and in each application of the Schur complement operator in Algorithm 1, we use plain CG which typically requires only a couple of iterations to converge.

5. Numerical results

In this section we demonstrate the efficiency and stability of our preconditioner on a number of test problems which are introduced in the following. All experiments were performed on a workstation with an Intel Xeon

E5-1620 v2 CPU (3.70GHz) and 16GB of memory. CPU timings were obtained using only a single computational thread to reduce non-deterministic effects due to MATLAB's built-in multithreading.

5.1. Test problems

As a first test problem, we consider the heat equation in $\mathbb{R}^d$: Let $\Omega = (0,1)^d \subset \mathbb{R}^d$ and $I = (0,1)$. Then the problem reads

Problem 1 (P1). *Find u such that*

$$\begin{aligned} \partial_t u - \Delta u &= f && \text{in } I \times \Omega, \\ u &= 0 && \text{on } I \times \partial\Omega, \end{aligned} \tag{5.1}$$

where the right hand side f is constructed in such a way that the exact solution is given by

$$u(t,\mathbf{x}) = \sin(10\pi t)\phi(\mathbf{x}) \quad \text{where} \quad \phi(\mathbf{x}) = \prod_{j=1}^{d} \mathbf{x}_j(1-\mathbf{x}_j). \tag{5.2}$$

Our second problem addresses the performance of the preconditioner when applied to anisotropic diffusion problems in 2d. We consider the 2d unit disk $\Omega = B_1(0) \subset \mathbb{R}^2$, a time interval $I = (0,1)$ and pose the following problem:

Problem 2 (P2). *Given a parameter $\eta > 0$, find u such that*

$$\begin{aligned} \partial_t u - \nabla \cdot (\boldsymbol{D}_\eta \nabla u) &= 0 && \text{in } I \times \Omega, \\ u &= 0 && \text{on } I \times \partial\Omega, \\ u(0,\cdot) &= u_0 && \text{on } \Omega, \end{aligned} \tag{5.3}$$

where

$$\boldsymbol{D}_\eta(\mathbf{x}) = \boldsymbol{R}_\theta \begin{pmatrix} 1 & 0 \\ 0 & \eta \end{pmatrix} \boldsymbol{R}_\theta^{-1} \quad \text{for} \quad \boldsymbol{R}_\theta = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix},$$

$\theta = 40°$ and the initial condition is given by $u_0(x,y) = 1 - x^2 - y^2$.

Remark: Varying the parameter η changes the anisotropy of the operator $\mathbf{D}_\eta$. A similar anisotropic operator was considered in the benchmark paper [14] (Test 3). The aim of problem **P2** is to study sensitivity of the preconditioner on anisotropies, mesh and time discretization parameters.

The last problem features an additional convective term and the aim is to study the performance of the preconditioner up to the convection dominated regime. The problem setting is inspired by a widely used benchmark for transport equations [19]. To this end, we consider the 2d unit square $\Omega = (0,1)^2$, time interval $I = (0, 2\pi)$ and the following problem:

Problem 3 (P3). *Given a parameter $\varepsilon > 0$ find u such that*

$$\begin{aligned} \partial_t u + \boldsymbol{w} \cdot \nabla u - \varepsilon \Delta u &= 0 && \text{in } I \times \Omega, \\ u &= 0 && \text{on } I \times \partial\Omega, \\ u(0,\cdot) &= u_0 && \text{on } \Omega, \end{aligned} \tag{5.4}$$

where $\boldsymbol{w}(x,y) = \begin{pmatrix} 0.5 - y \\ x - 0.5 \end{pmatrix}$ and the initial condition is a "smooth hump" given by

$$u_0(x,y) = \begin{cases} \frac{1+\cos(\pi r(x,y))}{4} & \text{for} \quad r(x,y) = \frac{1}{r_0}\sqrt{(x-x_0)^2 + (y-y_0)^2} \le 1, \\ 0 & \text{else}, \end{cases}$$

with $r_0 = 0.15$ and $(x_0, y_0) = (0.25, 0.5)$.

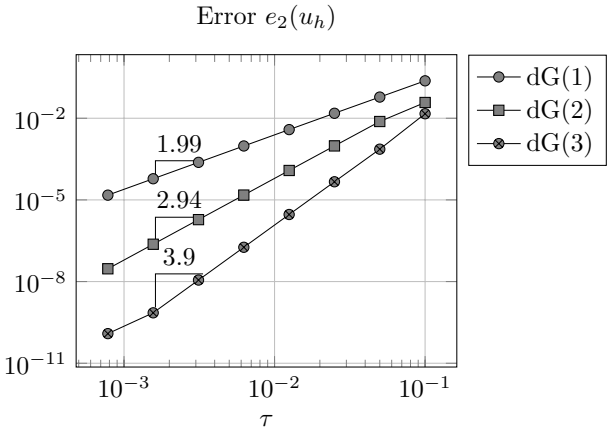

FIGURE 2. Convergence plot of the dG(k) methods for $k = 1, 2, 3$.

Notice: The operator in (5.4) is no longer symmetric. Therefore, this example is not covered by the analysis presented in Section 3.2. However, we will show that even when the diffusion coefficient $\varepsilon \to 0$, the preconditioner performs well.

5.2. Convergence results

The aim of this subsection is to verify our implementation. To this end, we consider problem **P1** in 1d, and discretize in space using quadratic polynomials. With this choice, the overall error will only be due to time discretization and errors due to solving the discrete systems (*e.g.* tolerances of the solvers).

Given a numerical solution u_h, let

$$e_2(u_h) := \left(\int_0^T ||\nabla(u - u_h)||^2_{L^2(\Omega)} \right)^{1/2} \tag{5.5}$$

denote its $L^2(H^1)$ error. To discretize in time, we employ the dG(k) methods for $k = 1, \ldots, 3$. Accordingly, convergence rates of $k+1$ with respect to τ are expected for $e_2(u_h)$. For fixed mesh size parameter $h = 1e-1$, the convergence behavior of the dG(k) methods with respect to $e_2(u_h)$ is plotted in Figure 2 confirming this expectation.

For the dG(k) and all combinations of $h \in \{10^{-l}, l = 1, \ldots, 3\}$ and $\tau \in \{0.1 \cdot 2^{-j}, j = 0, \ldots, 7\}$, we solve the space-time system (2.14) using

- the MATLAB operator \ (denoted by "direct solve"),
- the preconditioning strategy based on Algorithm 1 (denoted by "Algorithm 1"), with an exact evaluation of the preconditioner as described in the previous section.

As already indicated by Figure 2, for fixed h, all dG(k) methods and solution strategies exhibit the predicted convergence behavior as $\tau \to 0$. In the parameter range given above, we do not observe any differences between the direct solve and Algorithm 1 results for the dG(1) method. Recalling that a tolerance of $\text{tol} = 1e-10$ is used as termination criterion for Algorithm 1, we expect errors due to inexactly solving the linear system to become visible for $h, \tau \to 0$. The $e_2(u_h)$ errors for the dG(2) and dG(3) methods and smallest time step sizes $\tau \in \{0.1 \cdot 2^{-j}, j = 3, \ldots, 7\}$ are shown in Table 2. Clearly, the predicted convergence rates can be observed for all methods and solution strategies. However, for very small τ, the overall error $e_2(u_h)$ is influenced by the accuracy of the corresponding linear system solver. In this respect, the direct solver shows less sensitivity on the mesh size parameter h than our strategy based on Algorithm 1. We would like to emphasize that only

TABLE 2. Problem **P1**: Error $e_2(u_h)$ of the numerical solution obtained by the dG(2) and dG(3) methods, solved using Algorithm 1 and a direct solver for different combinations of h and τ.

	dG(2)				dG(3)			
	h = 1e-1							
	Algorithm 1		Direct solve		Algorithm 1		Direct solve	
τ	$e_2(u_h)$	EOC	$e_2(u_h)$	EOC	$e_2(u_h)$	EOC	$e_2(u_h)$	EOC
1.2500e-02	1.2049e-04	3.00	1.2049e-04	3.00	2.9086e-06	3.99	2.9086e-06	3.99
6.2500e-03	1.5075e-05	3.00	1.5075e-05	3.00	1.8196e-07	4.00	1.8196e-07	4.00
3.1250e-03	1.8848e-06	3.00	1.8848e-06	3.00	1.1375e-08	4.00	1.1375e-08	4.00
1.5625e-03	2.3561e-07	3.00	2.3561e-07	3.00	7.1109e-10	4.00	7.1101e-10	4.00
7.8125e-04	2.9452e-08	3.00	2.9452e-08	3.00	1.2244e-10	2.54	4.4438e-11	4.00
	h = 1e-2							
	Algorithm 1		Direct solve		Algorithm 1		Direct solve	
τ	$e_2(u_h)$	EOC	$e_2(u_h)$	EOC	$e_2(u_h)$	EOC	$e_2(u_h)$	EOC
1.2500e-02	1.2049e-04	3.00	1.2049e-04	3.00	2.9086e-06	3.99	2.9086e-06	3.99
6.2500e-03	1.5075e-05	3.00	1.5075e-05	3.00	1.8196e-07	4.00	1.8196e-07	4.00
3.1250e-03	1.8848e-06	3.00	1.8848e-06	3.00	1.1374e-08	4.00	1.1375e-08	4.00
1.5625e-03	2.3561e-07	3.00	2.3561e-07	3.00	7.1434e-10	3.99	7.1101e-10	4.00
7.8125e-04	2.9452e-08	3.00	2.9452e-08	3.00	1.6314e-10	2.13	4.4438e-11	4.00
	h = 1e-3							
	Algorithm 1		Direct solve		Algorithm 1		Direct solve	
τ	$e_2(u_h)$	EOC	$e_2(u_h)$	EOC	$e_2(u_h)$	EOC	$e_2(u_h)$	EOC
1.2500e-02	1.2049e-04	3.00	1.2049e-04	3.00	2.9086e-06	3.99	2.9086e-06	3.99
6.2500e-03	1.5075e-05	3.00	1.5075e-05	3.00	1.8203e-07	4.00	1.8196e-07	4.00
3.1250e-03	1.8848e-06	3.00	1.8848e-06	3.00	1.1445e-08	3.99	1.1375e-08	4.00
1.5625e-03	2.3563e-07	3.00	2.3561e-07	3.00	2.3065e-09	2.31	7.1106e-10	4.00
7.8125e-04	2.9970e-08	2.97	2.9452e-08	3.00	2.0278e-09	0.19	4.6148e-11	3.95

a fixed termination criterion of tol $= 1e-10$ for the relative residual was used in Algorithm 1, while different termination criteria for Algorithm 1 could be considered for such extreme cases.

5.3. Performance of the preconditioner

We now come to the main obective of our presentation, the efficient solution of the systems. We elaborate on the performance and stability of the proposed preconditioner.

*5.3.1. Performance of the preconditioner for problem **P1***

We first consider problem **P1** in one space dimension, discretize in space using piecewise quadratic polynomials and use different mesh sizes and time step sizes τ. To be precise, we choose $\tau \in \{10^{-l}, l = 1, \ldots, 4\}$ and $h \in \{0.2 \times 2^{-l}, l = 0, \ldots 4\}$. As time discretization schemes, we consider dG(1), dG(2) and dG(3).

Recall that the main numerical effort in the solution process lies in solving implicit Euler-like problems: for each block in equation (3.11) (corresponding to a pair of complex eigenvalues of the matrix $\mathbf{b}^{-1}\mathbf{g}$), the Schur complement formulation equation (4.13) is solved using Algorithm 1. In each iteration, an application of the preconditioner requires the solution of two implicit Euler-like problems. Once the essential unknown $\mathbf{w}_2$ of the block is obtained, one further solve with the mass matrix for the second unknown $\mathbf{w}_1$ is needed, as shown in (4.15). Also, for real eigenvalues in (3.11), only one additional solve is required. To get an idea of the overall effort, all Euler-like solves for the preconditioner matrix $(\mu_{\text{opt}}\mathbf{M} + \tau_n\mathbf{A})$ are recorded and also the maximum number of required CG iterations in Algorithm 1.

TABLE 3. Problem **P1**: *From top to bottom*: dG(1), dG(2), dG(3), each entry of table: maximum number of CG iterations of Algorithm 1 per block/total number of implicit Euler solves.

$\tau\backslash h$	2.00e-01	1.00e-01	5.00e-02	2.50e-02	1.25e-02
1.00e-01	5/10	5/10	5/10	5/10	5/10
1.00e-02	5/10	6/12	6/12	7/14	7/14
1.00e-03	5/10	6/12	6/12	6/12	6/12
1.00e-04	4/ 8	4/ 8	5/10	5/10	5/10
$\tau\backslash h$	2.00e-01	1.00e-01	5.00e-02	2.50e-02	1.25e-02
1.00e-01	5/11	7/15	7/15	7/15	7/15
1.00e-02	5/11	7/15	7/15	8/17	8/17
1.00e-03	5/11	6/13	7/15	7/14	7/15
1.00e-04	4/9	5/11	6/13	6/13	6/13
$\tau\backslash h$	2.00e-01	1.00e-01	5.00e-02	2.50e-02	1.25e-02
1.00e-01	5/18	8/26	8/26	8/26	8/26
1.00e-02	4/16	7/22	8/26	8/26	9/28
1.00e-03	5/18	7/22	7/22	7/22	7/22
1.00e-04	4/14	5/16	6/20	6/20	6/20

For each dG(k) and each combination of τ and h, these counts are shown in Table 3. As can be seen, typically only a couple of iterations are needed for Algorithm 1 to converge. For instance, for dG(2), whose discretization results in one 1×1 and one 2×2 block Algorithm 1 needs at most 8 iterations for the Schur complement PCG solve, each of which accounts for 2 implicit Euler-like solves, plus one solve for the 1×1 block, which gives a total of 17 required solves.

Comparing the total number of implicit Euler-like solves, from Table 3 we can also conclude that the preconditioned dG(2) method is "relatively cheap" compared to the dG(1) method and its successor dG(3). This is due to the fact that the block diagonal matrix $\mathbf{S}_h$ in (3.11) contains only one additional "cheap" 1×1 block compared to dG(1). For dG(3) on the other hand, $\mathbf{S}_h$ is made up of two "expensive" 2×2 blocks requiring two Schur complement solves.

We now elaborate on the performance of the preconditioner when applied to larger systems in 3d. To this end we discretize problem **P1** in space using linear finite elements defined on regular triangulations of the unit cube $\Omega = (0,1)^3 \subset \mathbb{R}^3$ consisting of $n_{\text{dof}} \in \{4096, 32\,768, 262\,144, 2\,097\,152\}$ nodes. We consider dG(k), $k = 0, \dots, 4$ and choose a constant time step size of $\tau = 1e-2$ corresponding to 100 time steps to discretize in time. Note that the fully coupled dG(k) space-time system consists of $(k+1) \times n_{\text{dof}}$ unknowns. The following strategies are used to solve this problem:

- direct solve of the full space-time system (2.14) using the MATLAB operator \,
- an iterative solution of the space-time system (2.14) using plain GMRES with restart after every 10th iteration and the relative stopping criterion of $1e-10$ that we use for Algorithm 1,
- GMRES as above, but with an incomplete LU factorization with no fill-in (ILU(0)) of the space-time system as preconditioner,
- The proposed preconditioning strategy based on Algorithm 1, where the application of the preconditioner $\mathbf{A}_{\text{opt}}^{-1} = (\mu_{\text{opt}}\mathbf{M} + \tau_n \mathbf{A})^{-1}$ is done using the MATLAB operator \, *i.e.* an "exact" evaluation of the preconditioner,
- Algorithm 1, where the preconditioner is solved approximatively by the algebraic multigrid solver AGMG, with two different stopping tolerances $\delta \in \{1e-10, 1e-5\}$, *i.e.* an "inexact" but efficient application of $\mathbf{A}_{\text{opt}}^{-1}$.

For each of these solution strategies and discretization parameters k and n_{dof} we measure the total CPU time needed to solve for 100 time steps and report our results in Table 4. Whenever Algorithm 1 is used we keep track of the maximum number of PCG iterations required in Algorithm 1 and report it in brackets.

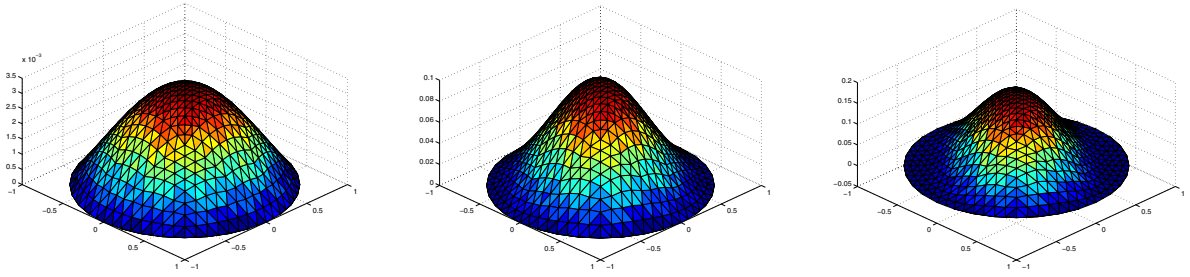

FIGURE 3. Solution of problem **P2** at $t = 1$ for three anisotropy parameters $\eta = 1, 1e-1, 1e-3$.

We first note that for our machine equipped with 16GB of memory, a direct solution of the systems generally fails for $n_{\text{dof}} \geq 262\,144$ due to insufficient memory (indicated by the symbol † in the table). This also prohibits the use of Algorithm 1 with an exact evaluation of the preconditioner for these problem sizes. Solution strategies based on direct solves of either the space-time system or the exact application of the preconditioner in Algorithm 1 prove to be very inefficient with increasing number of spatial degrees of freedom in general. Notice however that, as already seen in our one-dimensional experiments, the maximum number of PCG iterations required for Algorithm 1 to converge is insensitive with respect to the space discretization parameter n_{dof} and temporal degree k as predicted by the analysis.

For smaller problem sizes with $n_{\text{dof}} \leq 32\,768$ we observe that an iterative solution of the fully coupled space-time system using ILU preconditioned GMRES turns out to be acceptably efficient. However, we observe a strong dependency on the temporal polynomial degree k and the number of spatial degrees of freedom n_{dof} rendering this strategy rather inefficient for larger problems. This dependency is obviously even more pronounced when an unpreconditioned GMRES solver is used to solve the systems. For the largest system with $n_{\text{dof}} = 2097152$ an assembly of the dG(k) space-time system for $k > 0$ exceeded the system's memory limit and an iterative solution of the system using GMRES was not possible in these cases. Also, for $k = 0$ a solution with unpreconditioned GMRES would have taken longer than 12 hours and was aborted which is indicated by †† in the table.

Algorithm 1 in combination with the AGMG multigrid solver to approximate the operator $\mathbf{A}_{\text{opt}}^{-1}$ proves to be very efficient and clearly superior in our experiments. We observe an (almost) linear dependency of the required CPU time on the number of degrees of freedom (n_{dof}) and only a very mild dependency on the temporal degree k. Additional savings can be obtained from tuning the AGMG stopping criterion. Notice that compared to the exact application of the preconditioner, the maximum number of required PCG iterations in Algorithm 1 did (almost) not change, even when a relatively coarse termination tolerance of $\delta = 1e-5$ was used. This indicates that inexact solvers for the operator $\mathbf{A}_{\text{opt}}^{-1}$ show only little influence on the stability of the proposed preconditioning strategy.

*5.3.2. Performance of the preconditioner for problem **P2***

For the anisotropic test problem **P2**, we discretize the unit disk using quasi uniform meshes of mesh sizes $h \in \{0.1 \times 2^{-l}, l = 0, \dots 2\}$ and employ quadratic finite elements. The idea is to vary the anisotropy parameter $\eta \in \{1, 1e-1, 1e-3\}$ to study the influence of anisotropy on the performance of the preconditioner. Typical solution profiles to problem **P2** for different values of η are depicted in Figure 3.

We also vary step size $\tau \in \{10^{-l}, l = 1, \dots, 4\}$ and track the maximum total number of required implicit Euler solves which our preconditioner needs to solve the coupled system (3.11). Results for dG(k), $k \in \{1, 2, 3\}$ are shown in Table 5.

Clearly, the maximum number of required solves neither depends on the anisotropy parameter η nor time and space discretization parameters, corroborating the uniform *a priori* result (3.22).

TABLE 4. Problem **P1** (3d): CPU times (in seconds) needed to solve 100 time steps of the dG(k) methods, $k = 0, \ldots, 4$ on four refinement levels: solution of system (2.14) using MATLAB \ (first row), using plain GMRES and ILU preconditioned GMRES (second and third row), using Algorithm 1 with exact evaluation of preconditioner and using Algorithm 1 with inexact preconditioner realized by AGMG (last three rows). For methods based on Algorithm 1, the maximum number of required PCG iterations is shown in brackets.

	dG(0)				dG(1)			
Strategy\n_{dof}	4096	32 768	262144	2 097 152	4096	32 768	262 144	2 097 152
Direct solve	2.48	158.22	†	†	10.31	1030.71	†	†
GMRES	2.48	45.09	1739.88	††	7.09	141.86	4287.30	†
GMRES (ILU)	1.26	14.10	318.81	6616.65	2.40	37.53	813.99	†
Algorithm 1, exact	2.73 (0)	160.09 (0)	†	†	32.55 (6)	1917.66 (6)	†	†
Algorithm 1, $\delta = 1e-10$	1.04 (0)	9.83 (0)	93.32 (0)	922.27 (0)	10.10 (6)	89.56 (6)	1059.80 (7)	9848.30 (7)
Algorithm 1, $\delta = 1e-5$	0.83 (0)	7.79 (0)	76.50 (0)	676.42 (0)	8.94 (6)	78.44 (6)	868.19 (7)	7089.91 (7)

	dG(2)				dG(3)			
Strategy\n_{dof}	4096	32 768	262144	2 097 152	4096	32 768	262 144	2 097 152
Direct solve	28.18	3221.89	†	†	58.06	7485.93	†	†
GMRES	13.94	391.18	10406.42	†	21.15	635.69	22080.81	†
GMRES (ILU)	4.02	77.04	1527.15	†	6.25	122.11	2513.50	†
Algorithm 1, exact	36.89 (7)	2410.70 (8)	†	†	62.75 (7)	3590.68 (8)	†	†
Algorithm 1, $\delta = 1e-10$	11.65 (7)	111.19 (8)	1311.53 (8)	11 823.68 (8)	19.04 (7)	171.11 (8)	2032.90 (9)	19 052.17 (9)
Algorithm 1, $\delta = 1e-5$	12.16 (7)	102.88 (8)	1126.74 (8)	8786.58 (8)	18.43 (7)	171.14 (8)	1784.25 (9)	13 942.03 (9)

	dG(4)			
Strategy\n_{dof}	4096	32 768	262 144	2 097 152
Direct solve	103.50	13 975.47	†	†
GMRES	38.38	1053.40	36 439.17	†
GMRES (ILU)	8.68	176.75	4304.79	†
Algorithm 1, exact	72.47 (8)	4362.67 (9)	†	†
Algorithm 1, $\delta = 1e-10$	19.63 (8)	221.22 (9)	2292.01 (9)	20 927.26 (10)
Algorithm 1, $\delta = 1e-5$	19.42 (8)	213.95 (9)	2057.80 (9)	15 663.70 (10)

TABLE 5. Problem **P2**: *From top to bottom*: dG(1), dG(2), dG(3), each entry of table: maximum number of implicit Euler solves for anisotropy parameters $\eta = 1/\eta = 1e-1/\eta = 1e-3$.

$\tau\backslash h$	1.00e-01	5.00e-02	2.50e-02
1.00e-01	10/10/10	12/12/12	12/12/12
1.00e-02	8/10/12	10/12/12	10/12/12
1.00e-03	10/ 8/ 8	10/10/10	10/10/10
1.00e-04	6/ 6/ 6	8/ 6/ 6	8/ 8/ 8
$\tau\backslash h$	1.00e-01	5.00e-02	2.50e-02
1.00e-01	11/13/13	13/15/15	15/15/15
1.00e-02	11/15/15	13/15/15	13/15/15
1.00e-03	11/11/9	13/11/11	13/13/13
1.00e-04	7/ 7/ 7	9/9/ 7	9/9/9
$\tau\backslash h$	1.00e-01	5.00e-02	2.50e-02
1.00e-01	18/22/22	22/22/26	26/24/68
1.00e-02	20/20/20	20/22/22	22/22/22
1.00e-03	16/14/14	18/16/16	20/20/20
1.00e-04	12/12/12	14/12/12	14/14/14

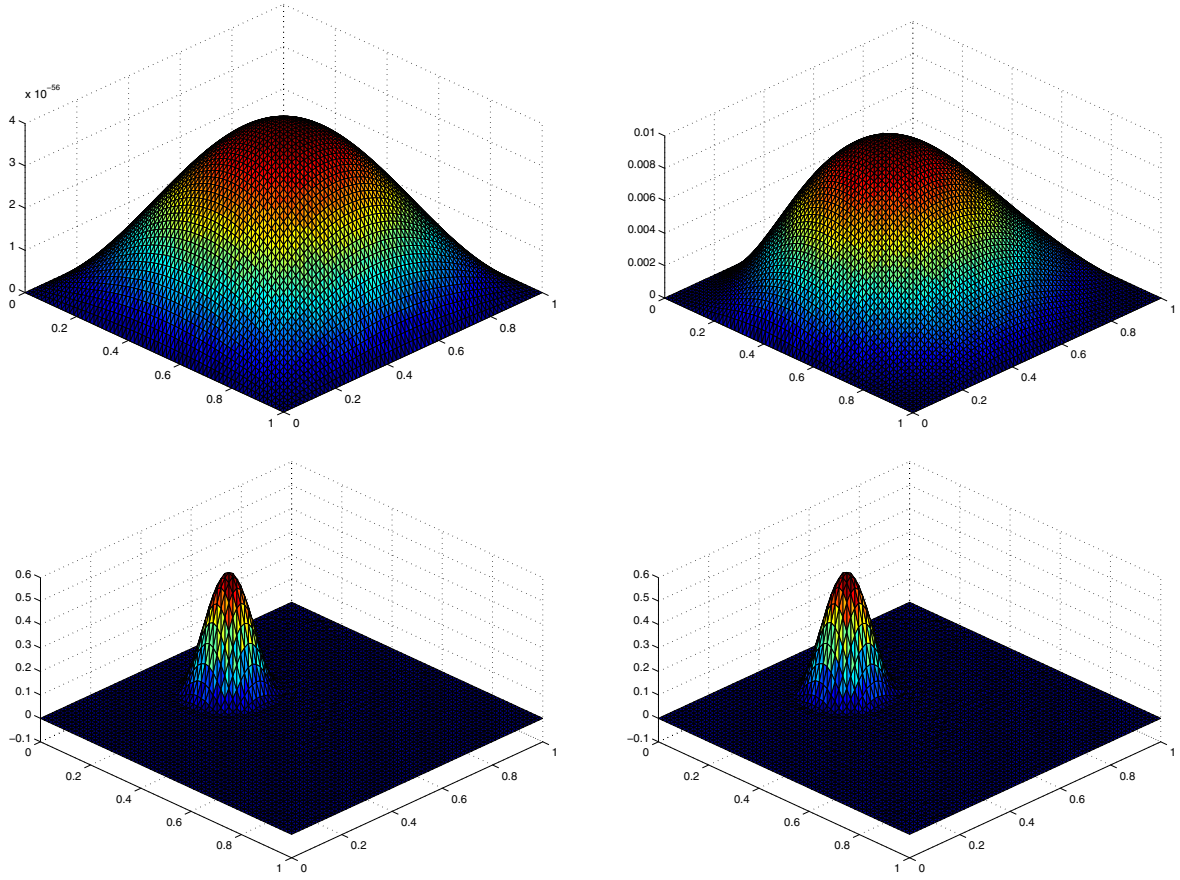

FIGURE 4. Solution of problem **P3** at $t = 2\pi$ for diffusion coefficients $\varepsilon = 1, 1e-2, 1e-6, 0$.

*5.3.3. Performance of the preconditioner for problem **P3***

Finally, we elaborate on the performance of the preconditioner when applied to convection dominated problems. In this case the analysis presented in Section 3 does not apply directly. To account for the loss of symmetry, we change Algorithm 1 which was based on CG in favor of BiCG, applied to the same left-right preconditioned operator (3.16). In contrast to CG, BiCG needs two applications of the preconditioner which makes it approximately twice as expensive per iteration. To allow for a "fair" comparison with CG in terms of the number of iterations required for Algorithm 1 to converge (which is essentially determined by the condition number of the preconditioned system), we record the maximum number of BiCG iterations in Algorithm 1 this time.

Just as for problem **P2**, we employ quadratic finite elements on uniform triangulations of the unit square for space discretization. For different mesh and time step sizes we vary the diffusion coefficient $\varepsilon \in \{1, 1e-2, 1e-6, 0\}$. The results are shown in Table 6, and a Figure corresponding to the solution at $t = 2\pi$ for different diffusion coefficients is shown in Figure 4.

We observe that the number of iterations does not show any particular dependency on both, problem and discretization parameters, even when diffusion is very small. Although this scenario was not covered by our analysis, this an indication that the proposed preconditioning strategy also works in the convection dominated regime.

TABLE 6. Problem **P3**: *From top to bottom*: dG(1), dG(2), dG(3), each entry of table: maximum number of BiCG iterations for diffusion coefficients $\varepsilon = 1/\varepsilon = 1e-2/\varepsilon = 1e-6/\varepsilon = 0$.

$\tau\backslash h$	1.00e-01	5.00e-02	2.50e-02
1.00e-01	4/4/5/5	4/4/6/6	4/4/5/5
1.00e-02	4/3/3/3	4/4/3/3	4/3/3/3
1.00e-03	4/3/2/2	4/3/2/2	4/3/2/2
$\tau\backslash h$	1.00e-01	5.00e-02	2.50e-02
1.00e-01	5/5/7/7	5/5/8/8	5/5/8/7
1.00e-02	4/4/3/3	4/4/3/3	4/4/5/5
1.00e-03	4/3/2/2	5/3/2/2	5/4/2/2
$\tau\backslash h$	1.00e-01	5.00e-02	2.50e-02
1.00e-01	5/5/7/7	5/5/9/9	5/5/9/9
1.00e-02	5/4/3/3	5/4/3/3	5/4/4/5
1.00e-03	5/3/2/2	6/3/2/2	5/4/2/2

6. Conclusion

We have developed an efficient yet conceptually simple preconditioner for the solution of systems arising from variational time discretization methods of arbitrary order. The main ingredients are a transformation of the system into real block diagonal form and a preconditioned Schur complement approach for the solution of the resulting 2×2 block systems. The preconditioner is based on an inexact factorization of the Schur complement operator consisting of implicit Euler-like problems. Consequently, the entire solution strategy reduces to solving simple Euler-like problems for which common efficient solution techniques can be recycled. Analysis shows that in case of a symmetric, positive operator A_h the preconditioned Schur complement operator has condition number ≤ 2 independent of all space and time discretization parameters. Numerical experiments indicate the robustness of our preconditioner.

References

[1] T. Akman and B. Karasözen, Variational time discretization methods for optimal control problems governed by diffusion–convection–reaction equations. *J. Comput. Appl. Math.* **272** (2014) 41–56.

[2] G. Akrivis, Ch. Makridakis and R.H. Nochetto, Galerkin and Runge–Kutta methods: unified formulation, a posteriori error estimates and nodal superconvergence. *Numer. Math.* **118** (2011) 429–456.

[3] E. Bänsch,, P. Morin and R.H. Nochetto, Preconditioning a class of fourth order problems by operator splitting. *Numer. Math.* **118** (2011) 197–228.

[4] A. Bonito, I. Kyza and R.H Nochetto, Time-discrete higher-order ale formulations: Stability. *SIAM J. Num. Anal.* **51** (2013) 577–604.

[5] A. Bonito, I. Kyza and R.H. Nochetto, A dG approach to higher order ale formulations in time. In Recent Developments in Discontinuous Galerkin Finite Element Methods for Partial Differential Equations. Springer (2014) 223–258.

[6] J.C. Butcher, On the implementation of implicit Runge–Kutta methods. *BIT Numer. Math.* **16** (1976) 237–240.

[7] K. Eriksson and C. Johnson, Adaptive finite element methods for parabolic problems i: A linear model problem. *SIAM J. Numer. Anal.* **28** (1991) 43–77.

[8] A. Ern and J.L. Guermond, Theory and practice of finite elements. Vol. 159. Springer Science & Business Media (2013).

[9] L.C. Evans, Partial differential equations. Vol. 19 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2nd edition (2010).

[10] R. Freund, On conjugate gradient type methods and polynomial preconditioners for a class of complex non-hermitian matrices. *Numer. Math.* **57** (1990) 285–312.

[11] R.W. Freund, Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices. *SIAM J. Sci. Stat. Comput.* **13** (1992) 425–448.

[12] G.H. Golub and Ch.F. Van Loan, Matrix computations. *Johns Hopkins Studies in the Mathematical Sciences*. Johns Hopkins University Press, Baltimore, MD, 4th edition (2013).

[13] E. Hairer and G. Wanner, Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems. Springer, second revised edition (1991).

[14] R. Herbin and F. Hubert, Benchmark on discretization schemes for anisotropic diffusion problems on general grids. In Finite volumes for complex applications V. Wiley (2008) 659–692.
[15] S. Hussain, F. Schieweck and S. Turek, Higher order Galerkin time discretizations and fast multigrid solvers for the heat equation. *J. Numer. Math.* **19** (2011) 41–61.
[16] L.O. Jay, Inexact simplified Newton iterations for implicit Runge–Kutta methods. *SIAM J. Numer. Anal.* **38** (2000) 1369–1388.
[17] L.O. Jay and Th. Braconnier, A parallelizable preconditioner for the iterative solution of implicit Runge–Kutta-type methods. *J. Comput. Appl. Math.* **111** (1999) 63–76.
[18] P. Lesaint and P.A. Raviart, On a Finite Element Method for Solving the Neutron Transport Equation. Univ. Paris VI, Labo. Analyse Numérique (1974).
[19] R.J. LeVeque, High-resolution conservative algorithms for advection in incompressible flow. *SIAM J. Numer. Anal.* **33** (1996) 627–665.
[20] Ch. Makridakis and R.H. Nochetto, A posteriori error analysis for higher order dissipative methods for evolution problems. *Numer. Math.* **104** (2006) 489–514.
[21] K.-A. Mardal, T.K. Nilssen and G.A. Staff, Order-optimal preconditioners for implicit Runge–Kutta schemes applied to parabolic pdes. *SIAM J. Sci. Comput.* **29** (2007) 361–375.
[22] A. Napov and Y. Notay, An algebraic multigrid method with guaranteed convergence rate. *SIAM J. Sci. Comput.* **34** (2012) A1079–A1109.
[23] Y. Notay, An aggregation-based algebraic multigrid method. *Electr. Trans. Numer. Anal.* **37** (2010) 123–146.
[24] Y. Notay, Aggregation-based algebraic multigrid for convection-diffusion equations. *SIAM J. Sci. Comput.* **34** (2012) A2288–A2316.
[25] Y. Notay, AGMG software and documentation. See http://homepages.ulb.ac.be/~ynotay/AGMG/ (2015).
[26] Th. Richter, A. Springer and B. Vexler, Efficient numerical realization of discontinuous Galerkin methods for temporal discretization of parabolic problems. *Numer. Math.* **124** (2013) 151–182.
[27] F. Schieweck and G. Matthies, Higher order variational time discretizations for nonlinear systems of ordinary differential equations. Otto-von-Guericke-Universität Magdeburg. Preprint (2011) 23.
[28] D. Schötzau and Ch. Schwab, Time discretization of parabolic problems by the hp-version of the discontinuous Galerkin finite element method. *SIAM J. Numer. Anal.* **38** (2000) 837–875.
[29] D. Schötzau and Ch. Schwab, hp-discontinuous galerkin time-stepping for parabolic problems. *C. R. Acad. Sci. Sér. I. Math.* **333** (2001) 1121–1126.
[30] G.A. Staff, K.A. Mardal and T.K. Nilssen, Preconditioning of fully implicit Runge-Kutta schemes for parabolic PDEs. *Model. Identif. Control* **27** (2006) 109.
[31] M. Stoll, One-shot solution of a time-dependent time-periodic PDE-constrained optimization problem. *IMA J. Numer. Anal.* (2013) drt019.
[32] M. Stoll and A. Wathen, All-at-once solution of time-dependent Stokes control. *J. Comput. Phys.* **232** (2013) 498–515.
[33] R. Temam, Navier-Stokes equations. Theory and numerical analysis. Vol. 2. North-Holland Publishing Co., Amsterdam-New York-Oxford (1977).
[34] V. Thomée, Galerkin Finite Element Methods for Parabolic Problems. Number 1054 in *Springer Lecture notes in Mathematics*. Springer, 2 edition (1984).
[35] Sh.W. Walker, Felicity: Finite element implementation and computational interface tool for you. MATLAB/C++, Tech. Report (2013).
[36] S. Weller and S. Basting, Efficient preconditioning of variational time discretization methods for parabolic partial differential equations. *ESAIM: M2AN* **49** (2015) 331–347.
[37] Th. Werder, K. Gerdes, D. Schötzau and Ch. Schwab, hp-discontinuous Galerkin time stepping for parabolic problems. *Comput. Methods Appl. Mech. Engrg.* **190** (2001) 6685–6708.