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

STEFFEN BASTING AND EBERHARD BÄNSCH

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.

1. Introduction

Using low order time discretization schemes like backward Euler or Crank–Nicolson is rather standard practise 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 [23, 9] 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 [14, 19].
- 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 [23].
- 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.

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 [6, 16, 1]. Moreover they are in particular well suited to discretize problems on time–dependent domains, see [2, 3].

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. [20, 21, 26] consider the dG(k) methods

(Steffen Basting) Institute of Applied Mathematics (LS III), TU Dortmund, Vogelpothsweg 8, 44227 Dortmund, Germany

(Eberhard Bänsch) Applied Mathematics III, Dept. of Mathematics, Cauerstr. 11, 91058 Erlangen, Germany

E-mail addresses: steffen.basting@math.tu-dortmund.de,baensch@math.fau.de.

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 [18] 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 [11].

At this point, it seems important to emphasize that variational time discretization schemes share many similarities with implicit Runge-Kutta methods, see for instance [1] 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 [9] of the Runge-Kutta coefficient matrix were discussed in [13, 12]. Mardal et al. analyze a block diagonal preconditioner for A-stable Runge-Kutta schemes in [17] 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 preceeding publication [22], 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 only *decoupled* 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 multi–grid 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 [5] for preconditioning fourth order (in space) problems. The main ingredient is an inexact factorization of the Schur complement of the 2×2 system for which uniform bounds with respect to space and time discretization parameters as well as the degree of the method can be proven. In this sense, the approach can be seen as a generalization of a similar Schur complement approach for the dG(1) method proposed in [25].

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*

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

Assume that $\mathbf{D}_{i,j}, \mathbf{w}_i \in L^\infty((0,T), L^\infty(\Omega)), i, j = 1, \dots, d$ and that the coefficient matrix $\mathbf{D}$ is symmetric and uniformly positive definite.

For the ease of notation we restrict ourselves to homogeneous Dirichlet boundary conditions, the extension to more general boundary conditions is straightforward.

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 $\mathbb{V}^* = H^{-1}(\Omega)$ and the bilinear form $a : \mathbb{V} \times \mathbb{V} \to \mathbb{R}$:

$$a(u, v) := \int_\Omega \mathbf{D} \nabla u \cdot \nabla v + \mathbf{w} \cdot \nabla u v \, \mathrm{d}x. \tag{1.2}$$

Furthermore, we define the parabolic function space

$$X := \{v \in L^2((0,T),\mathbb{V}) : \ \partial_t v \in L^2((0,T),\mathbb{V}^*)\}. \tag{1.3}$$

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

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

for all $v \in X$.

Here, $(\cdot,\cdot)$ denotes the L^2-inner product and $\langle\cdot,\cdot\rangle$ is the duality pairing on $\mathbb{V}$. Existence and uniqueness of a weak solution can be proved (see e.g. [7]). In particular, since $X \hookrightarrow C^0([0,T],L^2(\Omega))$, $v(0) \in L^2(\Omega)$ is well defined and the initial condition is also well defined in this sense.

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 Eq. (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

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

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 [23]:

For given $u_\tau(0) = u_0$ *and* $f \in L^2((0,T),\mathbb{V}^*)$, *find* $u_\tau \in \mathbb{P}_k^{\mathrm{dc}}$, *such that*

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

This method is known to be strongly A-stable (L-stable) (Chp. 4.1.3, Lemma 3 + 5 in [19]). 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 [23]).

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 $\{\varphi_{n,j} \in \mathbb{P}_k(I_n, \mathbb{R}), j = 1, \dots, k+1\}$ 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 concrete basis of $\mathbb{P}_k(I_n, \mathbb{V})$ has to be specified and the time integrals in Eq. (2.2) need to be replaced by numerical quadrature. To this end, let

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

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 Eq. (2.2) by numerical quadrature and using a (yet to be specified) basis $\{\varphi_{n,j}, j = 1, \dots, k+1\}$ 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 $n \in \{1, \dots N\}$. For given $u_\tau|_{I_{n-1}}$ from the previous time step (or initial data if $n = 1$), find $k+1$ coefficients $u_n^1, \dots, u_n^{k+1} \in \mathbb{V}$ such that the following equations hold for all $v \in \mathbb{V}$:

$$\begin{gathered} \sum_{j=1}^{k+1} \gamma_{i,j}(u_n^j, v) + \alpha_{i,j} a(u_n^j, v) = \\ (u_{n-1}^-, v)\varphi_{n,i}(t_{n-1}) + \mathcal{Q}_n[\langle f(t), v\rangle \varphi_{n,i}(t)] \qquad \text{for } 1 \le i \le k+1, \\ \text{where} \quad \gamma_{i,j} := \mathcal{Q}_n[\varphi'_{n,j}(t)\varphi_{n,i}(t)] + \varphi_{n,j}(t_{n-1})\varphi_{n,i}(t_{n-1}) \quad \text{and} \quad \alpha_{i,j} := \mathcal{Q}_n[\varphi_{n,j}(t)\varphi_{n,i}(t)]. \end{gathered} \tag{2.7}$$

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 Lemma 3.1). Following [19], 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 $t_{n,i} \in I_n, i = 1, \dots, k+1$. 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 Eq. (2.7) as a matrix valued problem on $\mathbb{V}_h$. To this end we introduce the operator-valued system matrices

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

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 $\mathbf{U}_h = \left(u_h^1, \quad \dots \quad , u_h^{k+1}\right)^T \in (\mathbb{V}_h)^{k+1}$ 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

$$\boxed{(\mathbf{G}_h + \mathbf{B}_h)\, \mathbf{U}_h = \mathbf{F}_h.} \tag{2.14}$$

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 $\mathbb{V}_h$:

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

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

$$\boxed{(\mathbf{g} \otimes I_h + \tau_n \mathbf{b} \otimes A_h)\, \mathbf{U}_h = \mathbf{F}_h,} \tag{3.2}$$

where the coefficients 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 Eq. (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 Eq. (3.2) by $\mathbf{b}^{-1}$ yields

$$\left(\mathbf{b}^{-1}\mathbf{g} \otimes I_h + \tau_n \mathbf{1} \otimes A_h\right) \mathbf{U}_h = \mathbf{b}^{-1}\mathbf{F}_h. \tag{3.3}$$

One key idea, well-known in the Runge-Kutta community (see for instance [9], Chapter IV.8) due to the pioneering work by Butcher [4], 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 1. *Matrix $\boldsymbol{b}^{-1}\boldsymbol{g}$ is diagonalizable.*

For convenience the eigenvalues $\lambda_1, \dots, \lambda_r, \lambda_{r+1}, \dots, \lambda_{k+1}$ of $\mathbf{b}^{-1}\mathbf{g}$ may be ordered such that the first r eigenvalues $\lambda_1, \dots, \lambda_r$ 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 $\{\lambda_{r+1}, \dots, \lambda_{k+1}\} = \left\{\lambda_{r+1}, \overline{\lambda}_{r+1}, \dots, \lambda_n, \overline{\lambda}_n\right\}$ for $n = \frac{k-r+1}{2}$, where $\Im(\lambda_i) > 0$ and $\Im(\overline{\lambda}_i) < 0$ for $r+1 \le i \le n$.

Then there exists a real valued transformation matrix $\mathbf{V} \in \mathbb{R}^{k+1\times k+1}$ such that

$$\mathbf{V}^{-1}\left(\mathbf{b}^{-1}\mathbf{g}\right)\mathbf{V} = \begin{pmatrix} \Lambda_1 & & \\ & \ddots & \\ & & \Lambda_n \end{pmatrix}, \tag{3.4}$$

is block diagonal with $\Lambda_i = \lambda_i$ for $i \le r$ and

$$\Lambda_i = \begin{pmatrix} \alpha_i & \beta_i \\ -\beta_i & \alpha_i \end{pmatrix}, \tag{3.5}$$

where $\alpha_i = \Re(\lambda_i)$ and $\beta_i = \Im(\lambda_i)$ for $r+1 \le i \le n$. 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

$$\mathbf{S}_h \mathbf{W}_h = \mathbf{V}^{-1}\mathbf{b}^{-1}\mathbf{F}_h, \tag{3.6}$$

where

$$\mathbf{S}_h = \begin{pmatrix} \lambda_1 I_h + \tau_n A_h & & & & & & & \\ & \ddots & & & & & & \\ & & \lambda_r I_h + \tau_n A_h & & & & & \\ & & & \alpha_{r+1} I_h + \tau_n A_h & \beta_{r+1} I_h & & & \\ & & & -\beta_{r+1} I_h & \alpha_{r+1} I_h + \tau_n A_h & & & \\ & & & & & \ddots & & \\ & & & & & & \alpha_n I_h + \tau_n A_h & \beta_n I_h \\ & & & & & & -\beta_n I_h & \alpha_n I_h + \tau_n A_h \end{pmatrix}. \tag{3.7}$$

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

$$\begin{pmatrix} \alpha I_h + \tau_n A_h & \beta I_h \\ -\beta I_h & \alpha I_h + \tau_n A_h \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}, \tag{3.8}$$

whose efficient solution will be subject of the next subsection. Note that due to the block diagonal structure, the solution of the sub-problems (3.8) can be done in parallel.

A crucial point is the sign of the real part α of the eigenvalues. This is clarified in the following lemma.

Lemma 3.1. *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 $\int_0^1 |v(\hat{t})|^2 d\hat{t} = 1$ and

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

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

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

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

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

Setting $w := v$ yields

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

□

Remarks:

- 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 $\mathbb{P}_k(I_n, \mathbb{V})$.
- The authors failed to prove that Assumption 1 is fulfilled for all $k \in \mathbb{N}$. However, computational tests clearly showed that the assumption is fulfilled for all $N \leq 50$, which is far beyond practical needs. A similar result was reported in [20], and the question of diagonalizability of the matrix $\mathbf{b}^{-1}\mathbf{g}$ seems to remain open.
- Note that Assumption 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 Eq. (3.4) by a real Schur transformation, i.e. a transformation into a block tridiagonal form, see [8]. Of course it is more convenient and more effective to work with a block diagonal structure.

3.2. **Solving the 2×2 block system by a preconditioned Schur complement formulation.** The solution of the 2×2 block system (3.8) 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

$$S_h w_2 = \beta f + (\alpha I_h + \tau_n A_h)g, \text{ where } S_h = (\alpha I_h + \tau_n A_h)^2 + \beta^2 I_h, \tag{3.9}$$

and then w_1 is determined from $(\alpha I_h + \tau_n A_h)w_1 = f - \beta w_2$. Clearly, the performance of the entire process will depend on how efficient the Schur complement equation (3.9) can be solved, and some remarks seem to be in order.

Remarks:

- 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 (3.8) does not possess this property since it has saddle point structure.

- S_h is a (discretized) fourth order differential operator, which means that the conditioning of problem (3.9) will be, in general, even worse than that of problem (3.8).
- An efficient solution strategy for (3.9) using iterative solvers will only be possible with a suitable preconditioner that will be constructed in this section.

The preconditioner for (3.9) is based on ideas developed for a class of fourth order problems presented and analyzed in [5]. 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 [25]. Note that an exact factorization for S_h is also possible, but leads to complex-valued equations, as pointed out in [18].

The preconditioner will be derived from and analyzed under the assumption that A_h is symmetric, positive definite. We will later show numerically that the performance of the preconditioner does not degenerate when this assumption is violated. Consider the symmetric left-right preconditioned operator

$$P_h(\mu) := (\mu I_h + \tau_n A_h)^{-1} S_h (\mu I_h + \tau_n A_h)^{-1}, \tag{3.10}$$

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

$$\sigma(\mu I_h + \tau_n A_h) = \{\mu + \lambda : 0 \leq \lambda \in \sigma(\tau_n A_h)\} \subset (\mu, \infty). \tag{3.11}$$

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

$$\tilde{\lambda}_\mu(\lambda) = \frac{(\alpha + \lambda)^2 + \beta^2}{(\mu + \lambda)^2}. \tag{3.12}$$

In order to highlight the role of the parameter μ let us first assume that $\mu = \alpha$. Then

$$\tilde{\lambda}_\alpha(\lambda) = 1 + \frac{\beta^2}{(\alpha + \lambda)^2} \tag{3.13}$$

and since $\tilde{\lambda}$ is monotonously decreasing in λ, we have

$$\sup_{\xi \in \sigma(A_h)} \tilde{\lambda}_\alpha(\xi) \leq \tilde{\lambda}_\alpha(0) = 1 + \frac{\beta^2}{\alpha^2}, \tag{3.14}$$

$$\inf_{\xi \in \sigma(A_h)} \tilde{\lambda}_\alpha(\xi) \geq \lim_{\lambda \to \infty} \tilde{\lambda}_\alpha(\lambda) = 1 \tag{3.15}$$

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

$$\boxed{\kappa(P_h(\alpha)) \leq 1 + \frac{\beta^2}{\alpha^2}.} \tag{3.16}$$

Note that this bound is 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 Figure 3.2) that for increasing polynomial degree k, the ratio $\frac{\beta^2}{\alpha^2}$ increases as well. Consequently, we aim at finding a parameter μ which resolves this dependency and leads to uniform bounds.

We first note that

$$\frac{d}{d\lambda} \tilde{\lambda}_\mu(\lambda) = \frac{2}{(\mu + \lambda)^3} \left[(\alpha + \lambda)(\mu - \alpha) - \beta^2 \right], \tag{3.17}$$

which means that $\tilde{\lambda}_\mu$ is monotonously decreasing in λ if $\mu \leq \alpha$. Therefore, in this case, $\kappa(P_h(\mu)) \leq \frac{\tilde{\lambda}_\mu(0)}{\lim_{\lambda \to \infty} \tilde{\lambda}_\mu(\lambda)} \leq \frac{\tilde{\lambda}_\mu(0)}{1} = \frac{\alpha^2 + \beta^2}{\mu^2}$, which is minimal if $\mu = \alpha$ and would result in the non optimal bound (3.16).

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

$$\frac{d}{d\lambda}\tilde{\lambda}_\mu(\lambda^*) = 0 \Leftrightarrow \lambda^* = \frac{\beta^2}{\mu - \alpha} - \alpha, \text{ and} \tag{3.18}$$

$$\lambda^* \geq 0 \Leftrightarrow \mu \leq \alpha + \frac{\beta^2}{\alpha}, \tag{3.19}$$

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

$$\tilde{\lambda}_\mu(0) = \frac{\alpha^2 + \beta^2}{\mu^2}, \quad \lim_{\lambda\to\infty} \tilde{\lambda}_\mu(\lambda) = 1 \quad \text{and } \tilde{\lambda}_\mu(\lambda^*) = \frac{\beta^2}{\beta^2 + (\mu - \alpha)^2} \leq 1. \tag{3.20}$$

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

$$\kappa(P_h(\mu)) \leq \frac{\max(1, \tilde{\lambda}_\mu(0))}{\tilde{\lambda}_\mu(\lambda^*)}. \tag{3.21}$$

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

$$\kappa(P_h(\mu)) \leq \frac{1}{\tilde{\lambda}_\mu(\lambda^*)} = 1 + \frac{(\mu - \alpha)^2}{\beta^2}. \tag{3.22}$$

For the second case,

$$\kappa(P_h(\mu)) \leq \frac{\tilde{\lambda}_\mu(0)}{\tilde{\lambda}_\mu(\lambda^*)} = \frac{\alpha^2 + \beta^2}{\beta^2}\frac{\beta^2 + (\mu - \alpha)^2}{\mu^2}, \tag{3.23}$$

which is monotonously decreasing in μ for $\mu^2 < \alpha^2 + \beta^2$. Both cases (3.22) and (3.23) yield that $\kappa(P_h(\mu))$ is minimized when

$$\mu^* = \sqrt{\alpha^2 + \beta^2}, \tag{3.24}$$

which finally gives

$$\kappa(P_h(\mu^*)) \leq 1 + \frac{(\mu^* - \alpha)^2}{\beta^2} = 2 - 2\frac{\alpha}{\beta^2}\left(\sqrt{\alpha^2 + \beta^2} - \alpha\right) \tag{3.25}$$

The remaining case $\mu \geq \alpha + \frac{\beta^2}{\alpha}$ leads to non optimal bounds, since from (3.17) one concludes that $\tilde{\lambda}_\mu$ is monotonously increasing in λ, and therefore

$$\kappa(P_h(\mu)) = \frac{\lim_{\lambda\to\infty} \tilde{\lambda}_\mu(\lambda)}{\tilde{\lambda}_\mu(0)} = \frac{\mu^2}{\alpha^2 + \beta^2} \geq 1 + \frac{\beta^2}{\alpha^2}.$$

For the optimal parameter μ^* we have thus obtained the following estimate:

Proposition 3.2. *Let A_h be symmetric and positive definite. For $\alpha \geq 0, \beta \in \mathbb{R}$ given, let $\mu_{\text{opt}} := \sqrt{\alpha^2 + \beta^2}$. Then the preconditioned operator $P_h(\mu_{\text{opt}})$ has condition number*

$$\boxed{\kappa(P_h(\mu_{\text{opt}})) \leq 2 - 2\frac{\alpha}{\beta^2}\left(\sqrt{\alpha^2 + \beta^2} - \alpha\right) \leq 2.} \tag{3.26}$$

For the sake of brevity, we will always write μ_{opt} to refer to the optimal value $\mu_{\text{opt}} = \sqrt{\alpha^2 + \beta^2}$ for each block in (3.6), and will not distinguish between μ_{opt} for different blocks, time discretization methods or temporal polynomial degrees k.

This subsection is finished by a numerical experiment, indicating that the bounds (3.16) and (3.26) 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

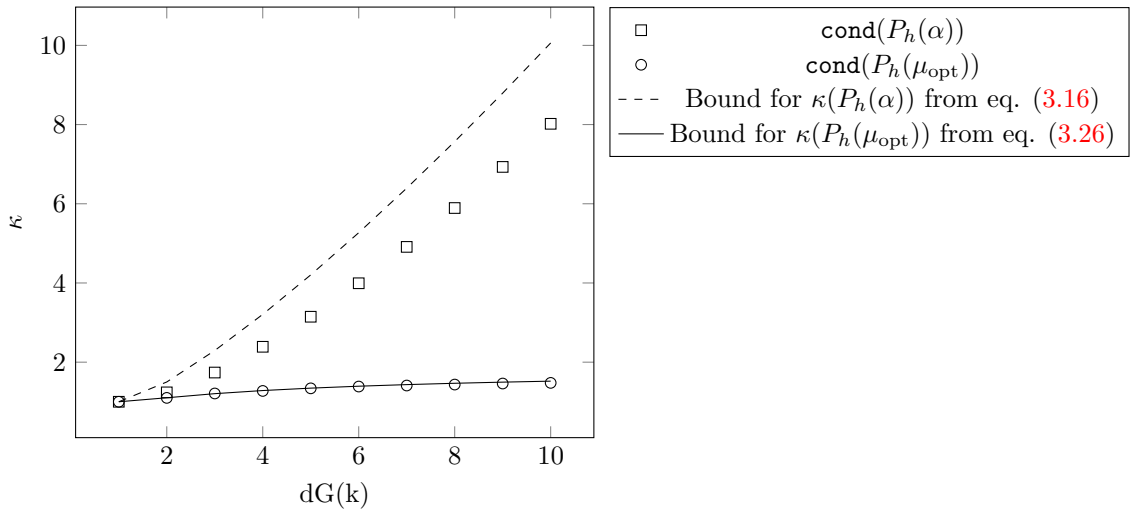

FIGURE 1. Condition numbers of the preconditioned operators $P_h(\alpha)$ and $P_h(\mu)$ for increasing polynomial degree k together with their theoretical bounds given in (3.16), (3.26).

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 (3.16), (3.26), and from a fully discretized 1d numerical example (i.e., a concrete discrete operator $\tau_n A_h$ corresponding to certain mesh size and time step size parameters, see Section 5.3.1 for details). The latter is done by using the MATLAB function `cond()`. The results are depicted in Figure 3.2. 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 chapter we elaborate on the choice of a concrete 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:

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

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

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

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

$$\mathbf{F}_i = (f, \varphi_i) \qquad i = 1, \dots, n_{\text{dof}} \tag{4.4}$$

$$\mathbf{G}_i = (g, \varphi_i) \qquad i = 1, \dots, n_{\text{dof}}, \tag{4.5}$$

respectively, and also, for a parameter θ, 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:

$$(I_h U_h, \varphi_l) = (U_h, \varphi_l) = (\mathbf{M}U)_l, \tag{4.7}$$
$$(A_h U_h, \varphi_l) = a(U_h, \varphi_l) = (\mathbf{A}U)_l, \tag{4.8}$$
$$((\theta I_h + \tau_n A_h)U_h, \varphi_l) = \theta(U_h, \varphi_l) + \tau_n a(U_h, \varphi_l) = (\mathbf{A}_\theta U)_l, \tag{4.9}$$
$$(f, \varphi_l) = \mathbf{F}_l, \tag{4.10}$$
$$(g, \varphi_l) = \mathbf{G}_l \tag{4.11}$$

for $l = 1, \dots, n_{\text{dof}}$, where $U \in \mathbb{R}^{n_{\text{dof}}}, U = (U_i)_{i=1,\dots,n_{\text{dof}}}$ is the coefficient vector of $U_h \in \mathbb{V}_{h,k}$.

To derive the matrix-vector equation that corresponds to the operator equation (3.9) in $\mathbb{V}_h$ we explain at first the isomorphism $r_m : \mathcal{L}(\mathbb{V}_h, \mathbb{V}_h) \to \mathbb{R}^{N\times N}, N = n_{\text{dof}}$, that assigns to a linear operator $A : \mathbb{V}_h \to \mathbb{V}_h$ its matrix representation $r_m(A) \in \mathbb{R}^{N\times N}$ in the following way (see [5]): 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 $U = r_v(U_h) \in \mathbb{R}^N$. Then, the matrix representation $r_m(A) \in \mathbb{R}^{N\times N}$ is defined by the property

$$r_v(AU_h) = r_m(A) r_v(U_h) \qquad \forall\, U_h \in \mathbb{V}_h.$$

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 by means of $\mathbf{A}_{kj} := (A\varphi_j, \varphi_k)$ for all $k, j = 1, \dots, N$, we can show that $r_m(A) = \mathbf{M}^{-1}\mathbf{A}$. For two operators A_1, A_2 with corresponding matrices $\mathbf{A}_1, \mathbf{A}_2$ we get $r_m(A_1 A_2) = \mathbf{M}^{-1}\mathbf{A}_1\mathbf{M}^{-1}\mathbf{A}_2$. Now, using the fact that $r_v(g) = \mathbf{M}^{-1}\mathbf{G}$ and the analog for f, we get that the operator equation (3.9) for $U_h \in \mathbb{V}_h$ is equivalent to the following matrix-vector equation for the nodal vector $W_2 = r_v(w_2) \in \mathbb{R}^N$:

$$\left(\mathbf{M}^{-1}\mathbf{A}_\alpha\mathbf{M}^{-1}\mathbf{A}_\alpha + \beta^2\mathbf{I}\right) W_2 = \beta\mathbf{M}^{-1}\mathbf{F} + \mathbf{M}^{-1}\mathbf{A}_\alpha\mathbf{M}^{-1}\mathbf{G} \tag{4.12}$$

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

$$\left(\mathbf{A}_\alpha\mathbf{M}^{-1}\mathbf{A}_\alpha + \beta^2\mathbf{M}\right) W_2 = \beta\mathbf{F} + \mathbf{A}_\alpha\mathbf{M}^{-1}\mathbf{G} \tag{4.13}$$

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

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

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.

Remark: The main numerical effort of Algorithm 1 lies in the evaluation of the preconditioner $(\mu_{\text{opt}}\mathbf{M} + \tau_n\mathbf{A})^{-1}$. This evaluation corresponds to solving one step of the implicit Euler method with time step size $\frac{\tau_n}{\mu_{\text{opt}}}$. 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.

4.2. **Implementation.** The implementation was done using the MATLAB based finite element toolbox FELICITY [24]. 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 Alg. 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 3.9 was solved using Alg. 1.

As termination criterion for Alg. 1, we prescribed a relative tolerance tol $= 1e-10$. The application of the preconditioner (i.e. evaluation of $\mathbf{A}_{\text{opt}}^{-1}$ and $\mathbf{M}^{-1}$ in Alg. 1) was done using the MATLAB operator \.

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 x_0
- $\omega_{\text{opt}} \leftarrow \sqrt{\alpha^2 + \beta^2}$ ▷ Compute optimal parameter for preconditioner
- $x \leftarrow x_0$ ▷ 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
- $b \leftarrow \left(\beta \mathbf{F} + \mathbf{A}_\alpha \mathbf{M}^{-1} \mathbf{G}\right)$ ▷ Compute right-hand side
- $r_0 \leftarrow r \leftarrow \mathbf{S}x - b$ ▷ Compute initial residual
- $h \leftarrow \mathbf{A}_{\text{opt}}^{-1} \mathbf{M} \mathbf{A}_{\text{opt}}^{-1} r_0$ ▷ Compute preconditioned residual
- $d \leftarrow h$
- **loop**
  - $\beta \leftarrow \frac{1}{\langle r, h \rangle}$
  - $z \leftarrow \mathbf{S}d$ ▷ Apply operator
  - $\alpha \leftarrow \frac{\langle r, h \rangle}{\langle d, z \rangle}$
  - $x \leftarrow x + \alpha d$ ▷ Update solution
  - $r \leftarrow r - \alpha z$ ▷ Update residual
  - **return** if $\|r\| / \|r_0\| < \text{tol}$
  - $h \leftarrow \mathbf{A}_{\text{opt}}^{-1} \mathbf{M} \mathbf{A}_{\text{opt}}^{-1} r$ ▷ Compute preconditioned residual
  - $\beta \leftarrow \beta \langle r, h \rangle$
  - $d \leftarrow h + \beta d$ ▷ Update search direction
- **end loop**
- **return** x ▷ $W_2 \leftarrow x$, solution to (4.13)

5. Numerical results

In this chapter we demonstrate the efficiency and stability of our preconditioner on a number of test problems which are introduced in the following.

5.1. **Test problems.** As a first test of our implementation, we consider the following 1d problem with time independent coefficients: Let $\Omega = (0,1) \subset \mathbb{R}$ and $I = (0,1)$.

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

$$f(t,x) := 10\pi \cos(10\pi t) x(1-x) + 2\sin(10\pi t). \tag{5.2}$$

The exact solution to this problem is given by

$$u(t,x) = \sin(10\pi t) x(1-x). \tag{5.3}$$

Remark: Note that for fixed t, both solution and right hand side are contained in the ansatz space if second order polynomials are used for space discretization. This means that no spatial error contribution is to be expected, and the error of the fully discrete solution will be due to time discretization only.

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 $\delta > 0$, find u such that*

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

where

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

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

Remark: Varying the parameter δ changes the anisotropy of the operator $\mathbf{D}_\delta$. A similar anisotropic operator was considered in the benchmark paper [10] (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 [15]. 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*

$$\begin{aligned} \partial_t u + \boldsymbol{w} \cdot \nabla u - \varepsilon \Delta u &= 0 && \textit{in } I \times \Omega, \\ u &= 0 && \textit{on } I \times \partial\Omega, \\ u(0, \cdot) &= u_0 && \textit{on } \Omega, \end{aligned} \tag{5.5}$$

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*

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

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

Remark: Note that the operator in (5.5) is no longer symmetric. Consequently, the analysis presented in Section 3.2 does not apply directly in this case. However, we will show that even when $\delta \to 0$, the preconditioner performs well.

5.2. **Convergence results.** We start by evaluating the correctness of our implementation. To this end the following notion for the numerical errors is introduced. Let u_h be a numerical solution. Define the $L^2(H^1)$ error

$$e_2(u_h) := \left(\int_0^T ||\nabla(u - u_h)||^2_{L^2(\Omega)} \right)^{1/2} \tag{5.6}$$

and the discrete $l^\infty(L^2)$ error at the nodal points $\{t_i\}$, $1 \leq i \leq N$,

$$e_\infty(u_h) := \max_n \|u(t_n) - u_h^n\|_{L^2(\Omega)}. \tag{5.7}$$

Throughout this section, uniform time steps $\tau_n = \tau$ are used on each time interval I_n. First consider problem **P1** discretized by polynomial elements of second order in space. With this choice, as mentioned above, the discretization error will only be due to time discretization and not space discretization.

For time discretization we employ dG(k) for $k = 0, \ldots 4$. Accordingly, convergence rates of $k + 1$ in the e_2 norm and super convergence rates of $2k + 1$ in the e_∞ norm are expected. On a relatively coarse mesh with mesh size $h = 0.1$, we vary the time step size τ and measure the error quantities. The results are depicted in Figure 5.2

Clearly, one observes the predicted convergence rates of $k + 1$ in the e_2 norm. The same holds for the predicted super convergence rate of $2k + 1$ in the e_∞ norm. For e_∞ the presentation is restricted to $0 \leq k \leq 2$, since for polynomial degrees $k \geq 3$ the error gets into the range of the machine precision. This simple example impressively demonstrates the power of the dG method.

5.3. **Computational costs.**

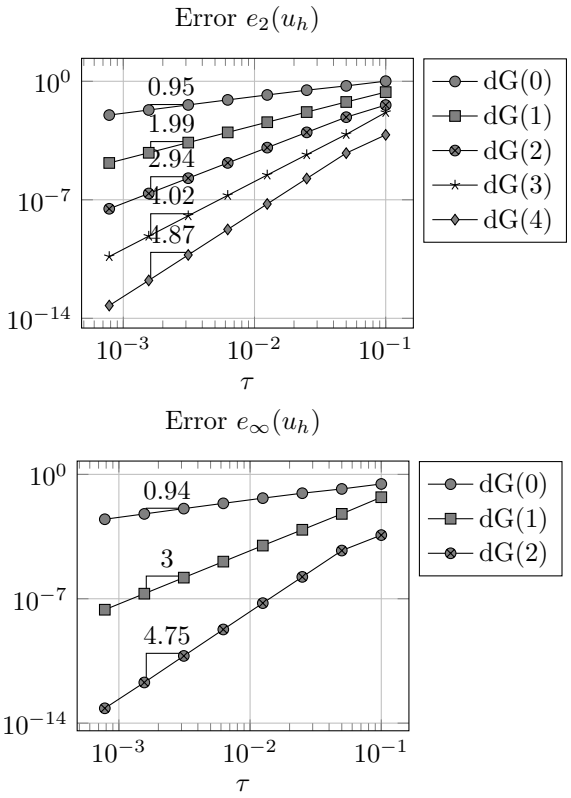

FIGURE 2. Convergence behavior of the dG(k) method in the e_2 (left) and e_∞ norms (right).

5.3.1. *Performance of the preconditioner for problem **P1**.* We now come to the main obective of our presentation, the efficient solution of the systems. We elaborate on the performance of the preconditioner. To this end, we again consider problem **P1**, 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 $h \in \{0.2 \cdot 2^{-l}, l = 0, \ldots 4\}$. 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 Eq. (3.6) (corresponding to a pair of complex eigenvalues of the matrix $\mathbf{b}^{-1}\mathbf{g}$), the preconditioned Schur complement formulation Eq. (3.10) 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 w_2 of the block is obtained, one further solve for the first unknown w_1 is needed. Also, for real eigenvalues in (3.6), only one additional solve is required. To get an idea of the overall effort, all Euler-like solves are therefore recorded and also the maximum number of required CG iterations in Algorithm 1.

For each dG(k) and each combination of τ and h, these counts are shown in Table 1. As can be seen, typically only a couple of iterations are needed for Algorithm 1 to converge. For instance, for dG(1), whose discretization results in one 2×2 block needs at most 7 iterations for the Schur complement PCG solve, each of which accounts for 2 implicit Euler-like solves, plus one solve for the second unknown, which gives a total of 15 solves.

Comparing the total number of implicit Euler-like solves, from Table 1 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 (3.6) 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.

$\tau\backslash h$	2.00e-01	1.00e-01	5.00e-02	2.50e-02	1.25e-02
1.00e-01	5/11	5/11	5/11	5/11	5/11
1.00e-02	5/11	6/13	6/13	7/15	7/15
1.00e-03	5/11	6/13	6/13	6/13	6/13
1.00e-04	4/ 9	4/ 9	5/11	5/11	5/11

$\tau\backslash h$	2.00e-01	1.00e-01	5.00e-02	2.50e-02	1.25e-02
1.00e-01	5/12	7/16	7/16	7/16	7/16
1.00e-02	5/12	7/16	7/16	8/18	8/18
1.00e-03	5/12	6/14	7/16	7/16	7/16
1.00e-04	4/10	5/12	6/14	6/14	6/14

$\tau\backslash h$	2.00e-01	1.00e-01	5.00e-02	2.50e-02	1.25e-02
1.00e-01	5/20	8/28	8/28	8/28	8/28
1.00e-02	4/18	7/24	8/28	8/28	9/30
1.00e-03	5/20	7/24	7/24	7/24	7/24
1.00e-04	4/16	5/18	6/22	6/22	6/22

TABLE 1. 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.

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 $h \in \{0.1 \cdot 2^{-l}, l = 0, \dots 2\}$ and employ quadratic finite elements. The idea is to vary the anisotropy parameter $\delta \in \{1, 1e-1, 1e-3\}$ to study the influence of anisotropy on the performance of the preconditioner. Typical solution profiles to problem **P2** for different values of δ are depicted in Figure 3.

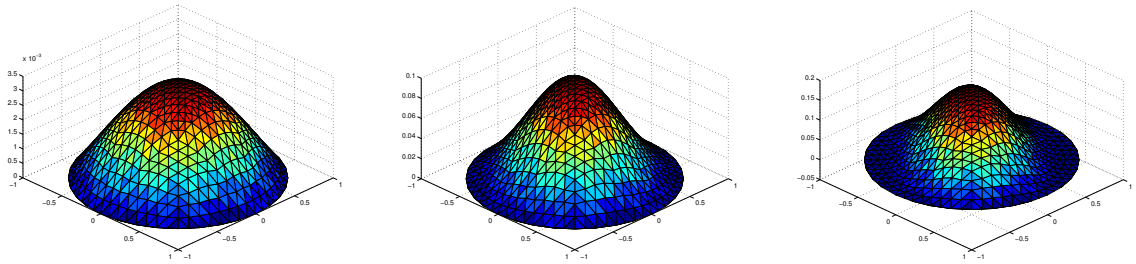

FIGURE 3. Solution of problem **P2** at $t = 1$ for the three anisotropy parameters $\delta = 1, 1e-1, 1e-3$.

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 (3.6). Results for dG(k), $k \in \{1, 2, 3\}$ are shown in Table 2.

Clearly, the maximum number of required solves neither depends on the anisotropy parameter δ nor time and space discretization parameters, corroborating the uniform a priori result (3.16).

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 Alg. 1 which was based on CG in favor of BiCG, applied to the same left-right preconditioned operator (3.10). In contrast to CG, BiCG needs two applications of the preconditioner which makes it approximately twice as expensive. To allow for a "fair" comparison with CG in terms of the number of iterations required for Alg. 1 to converge (which is essentially determined by

$\tau\backslash h$	1.00e-01	5.00e-02	2.50e-02
1.00e-01	11/11/11	13/13/13	13/13/13
1.00e-02	9/11/13	11/13/13	11/13/13
1.00e-03	11/ 9/ 9	11/11/11	11/11/11
1.00e-04	7/ 7/ 7	9/ 7/ 7	9/ 9/ 9

$\tau\backslash h$	1.00e-01	5.00e-02	2.50e-02
1.00e-01	12/14/14	14/16/16	16/16/16
1.00e-02	12/16/16	14/16/16	14/16/16
1.00e-03	12/12/10	14/12/12	14/14/14
1.00e-04	8/ 8/ 8	10/10/ 8	10/10/10

$\tau\backslash h$	1.00e-01	5.00e-02	2.50e-02
1.00e-01	20/24/24	24/24/28	28/26/28
1.00e-02	22/22/22	22/24/24	24/24/24
1.00e-03	18/16/16	20/18/18	22/22/22
1.00e-04	14/14/14	16/14/14	16/16/16

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

$\tau\backslash h$	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

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

the condition number of the preconditioned system), we record the maximum number of BiCG iterations in Alg. 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 3, 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. Clearly, this indicates that the proposed preconditioning strategy also works in the convection dominated regime.

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 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

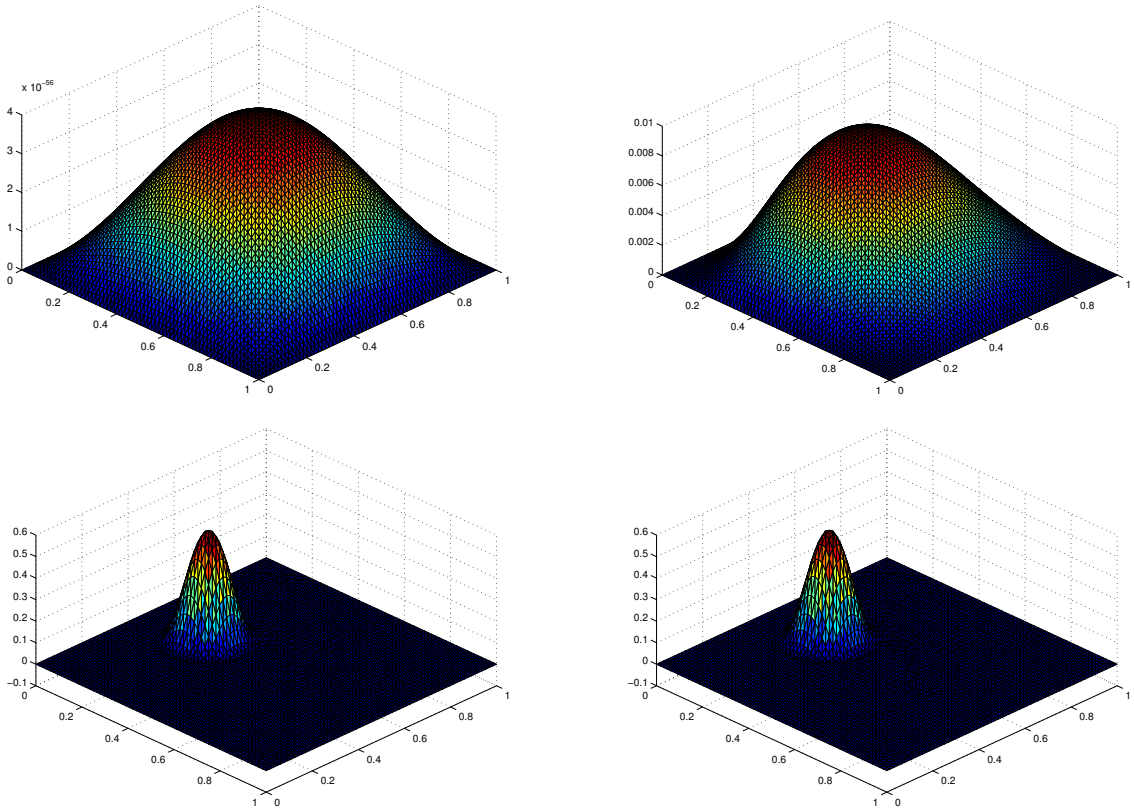

FIGURE 4. Solution of problem **P3** at $t = 2\pi$ for the diffusion coefficients $\varepsilon = 1, 1e-2, 1e-6, 0$.

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] Akrivis, G., Makridakis, C., Nochetto, R.H.: Galerkin and Runge–Kutta methods: unified formulation, a posteriori error estimates and nodal superconvergence. Numerische Mathematik **118**(3), 429–456 (2011)

[2] Bonito, A., Kyza, I., Nochetto, R.H.: Time-discrete higher-order ale formulations: Stability. SIAM Journal on Numerical Analysis **51**(1), 577–604 (2013)

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

[4] Butcher, J.C.: On the implementation of implicit Runge–Kutta methods. BIT Numerical Mathematics **16**(3), 237–240 (1976)

[5] Bänsch, E., Morin, P., Nochetto, R.H.: Preconditioning a class of fourth order problems by operator splitting. Numerische Mathematik **118**, 197–228 (2011). DOI 10.1007/s00211-010-0333-4

[6] Eriksson, K., Johnson, C.: Adaptive finite element methods for parabolic problems i: A linear model problem. SIAM Journal on Numerical Analysis **28**(1), 43–77 (1991)

[7] Evans, L.C.: Partial differential equations, *Graduate Studies in Mathematics*, vol. 19, second edn. American Mathematical Society, Providence, RI (2010)

[8] Golub, G.H., Van Loan, C.F.: Matrix computations, fourth edn. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD (2013)

[9] Hairer, E., Wanner, G.: Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems., second revised edition edn. Springer (1991)
[10] Herbin, R., Hubert, F.: Benchmark on discretization schemes for anisotropic diffusion problems on general grids. In: Finite volumes for complex applications V, pp. 659–692. Wiley (2008)
[11] Hussain, S., Schieweck, F., Turek, S.: Higher order Galerkin time discretizations and fast multigrid solvers for the heat equation. Journal of Numerical Mathematics **19**(1), 41–61 (2011)
[12] Jay, L.O.: Inexact simplified Newton iterations for implicit Runge–Kutta methods. SIAM Journal on Numerical Analysis **38**(4), 1369–1388 (2000)
[13] Jay, L.O., Braconnier, T.: A parallelizable preconditioner for the iterative solution of implicit Runge–Kutta-type methods. Journal of computational and applied mathematics **111**(1), 63–76 (1999)
[14] Lesaint, P., Raviart, P.: On a Finite Element Method for Solving the Neutron Transport Equation. Univ. Paris VI, Labo. Analyse Numérique (1974)
[15] LeVeque, R.J.: High-resolution conservative algorithms for advection in incompressible flow. SIAM Journal on Numerical Analysis **33**(2), 627–665 (1996)
[16] Makridakis, C., Nochetto, R.H.: A posteriori error analysis for higher order dissipative methods for evolution problems. Numerische Mathematik **104**(4), 489–514 (2006)
[17] Mardal, K.A., Nilssen, T.K., Staff, G.A.: Order-optimal preconditioners for implicit Runge–Kutta schemes applied to parabolic pdes. SIAM Journal on Scientific Computing **29**(1), 361–375 (2007)
[18] Richter, T., Springer, A., Vexler, B.: Efficient numerical realization of discontinuous Galerkin methods for temporal discretization of parabolic problems. Numerische Mathematik pp. 1–32 (2012)
[19] Schieweck, F., Matthies, G.: Higher order variational time discretizations for nonlinear systems of ordinary differential equations. Preprint 23/2011, Otto-von-Guericke-Universität Magdeburg (2011)
[20] Schötzau, D., Schwab, C.: Time discretization of parabolic problems by the hp-version of the discontinuous Galerkin finite element method. SIAM Journal on Numerical Analysis **38**(3), 837–875 (2000). DOI 10.1137/S0036142999352394
[21] Schötzau, D., Schwab, C.: hp-discontinuous galerkin time-stepping for parabolic problems. Comptes Rendus de l'Académie des Sciences - Series I - Mathematics **333**(12), 1121 – 1126 (2001). DOI 10.1016/S0764-4442(01)02186-3
[22] Staff, G.A., Mardal, K.A., Nilssen, T.K.: Preconditioning of fully implicit Runge-Kutta schemes for parabolic PDEs. Modeling Identification and Control **27**(2), 109 (2006)
[23] Thomée, V.: Galerkin Finite Element Methods for Parabolic Problems, 2 edn. No. 1054 in Springer Lecture notes in Mathematics. Springer (1984)
[24] Walker, S.W.: Felicity: Finite element implementation and computational interface tool for you (2013). URL http://felicity-finite-element-toolbox.googlecode.com/files/FELICITY_Ver0.915.pdf
[25] Weller, S., Basting, S.: Efficient preconditioning of variational time discretization methods for parabolic partial differential equations. ESAIM: M2AN **49**(2), 331–347 (2015). DOI 10.1051/m2an/2014036
[26] Werder, T., Gerdes, K., Schötzau, D., Schwab, C.: hp-discontinuous Galerkin time stepping for parabolic problems. Computer methods in applied mechanics and engineering **190**(49), 6685–6708 (2001)